

‘+.ly’:

0.1 Introduction

This document presents a feature test for LilyPond 2.0.3. When the text correspond with the shown notation, we consider LilyPond Officially BugFree (tm). This document is intended for finding bugs, and documenting bugfixes.

TODO: order of tests (file names!), test only one feature per test. Smaller and neater tests.

‘accidental-cautionary.ly’:

Cautionary accidentals are indicated using either parentheses (default) or smaller accidentals.



‘accidental-double.ly’:

If two forced accidentals happen at the same time, only one sharp sign is printed.



‘accidental-ledger.ly’:

Ledger lines are shortened when there are accidentals.



‘accidental-octave.ly’:

This shows how accidentals in different octaves are handled. (DOCME)





‘accidental-single-double.ly’:

A sharp sign after a double sharp sign, as well as a flat sign after a double flat sign is automatically prepended with a natural sign.



gisis’

gis’

gis’

ges’

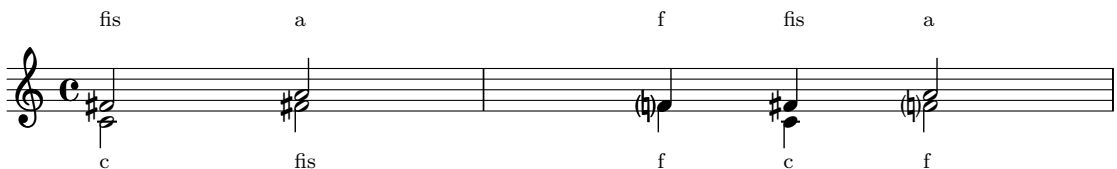
‘accidental-unbroken-tie-spacing.ly’:

Tied accidentaled notes (which cause reminder accidentals) don’t wreak havoc in the spacing when unbroken.



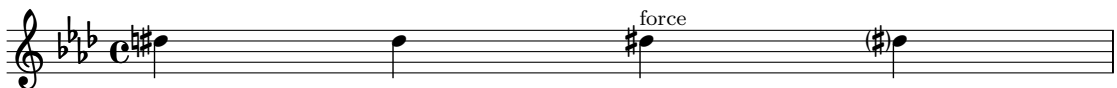
‘accidental-voice.ly’:

This shows how modern cross voice auto cautionary accidentals are handled. The first two fisses get accidentals because they belong to different voices. The first f gets cautionary natural because of previous measure. The last f gets cautionary natural because fis was only in the other voice.



‘accidental.ly’:

Accidentals work: the second note does not get a sharp. The third and fourth show forced and courtesy accidentals.



dis”

dis”

dis”

dis”

‘accidentals.ly’:

This shows how accidentals are handled.



d dis dis dis d d dis disisd dis d des disisdisd dis desesd dis dis dis disd dis



dis dis cis c c cis cisis cis c ces cisis c cis ceses c cis cis cis cis cis

`‘ambitus.ly’:`

Ambituses indicate pitch ranges for voices.

By default, the ambitus grob is put before the clef. You can control this behaviour through the `breakAlignOrder` property of the score context by redefining the order.

The shape of the note heads to use can be changed via the `note-head-style` property, which holds the glyph name of the note head. The vertical line between the upper and lower head can be switched on or off via the `join-heads` property.



`‘apply-context.ly’:`

With `\applycontext`, `\properties` can be modified procedurally. Applications include: checking bar numbers, smart octavation.

This example prints a bar-number during processing on stdout.



`‘apply-output.ly’:`

The `\applyoutput` expression is the most flexible way to tune properties for individual grobs. Here, the layout of a note head is changed depending on its vertical position.



`‘arpeggio-bracket.ly’:`

A square bracket on the left indicates that the player should not arpeggiate the chord.



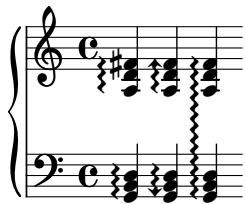
`‘arpeggio-collision.ly’:`

Arpeggio stays clear of accidentals and flipped note heads.



`‘arpeggio.ly’:`

Arpeggios are supported, both cross-staff and broken single staff.



`‘auto-beam-bar.ly’:`

No auto beams will be put over (manual) repeat bars.



`‘auto-beam-triplet.ly’:`

Automatic beaming is also done on triplets.



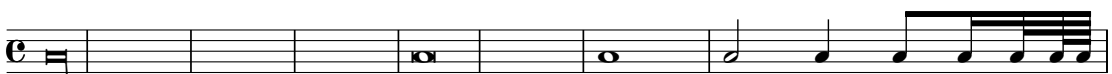
`‘auto-beam-tuplets.ly’:`

Triplet-spanner should not put (visible) brackets on beams even if they’re auto generated.



`‘auto-beam.ly’:`

Test automatic beamer: the last measure should have a single beam.



`‘auto-change.ly’:`

Auto change piano staff switches voices between up and down staves automatically rests are switched along with the coming note. When central C is reached, we don’t switch (by default).



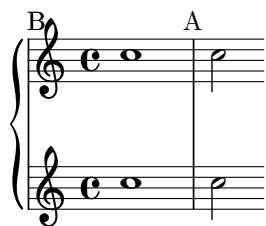
`'bar-number.ly':`

Bar number settable and padding adjustable. Bar numbers start counting after the anacrusis. The padding should be increased, to prevent clashes at the start of the line.



`'bar-scripts.ly':`

Markings that are attached to (invisible) barlines are delicate: they are attached to the rest of the score without the score knowing it. Consequently, they fall over often.



`'beam-auto-knee.ly':`

Automatic kneeling. A knee is made when a horizontal beam fits in a gap between note heads that is larger than a predefined threshold.



`'beam-break.ly':`

Beams can be printed across line breaks if forced.



`'beam-center-slope.ly':`

Simple beams on middle staffline are allowed to be slightly sloped, even if the notes have ledgers. Beams reaching beyond middle line can have bigger slope.



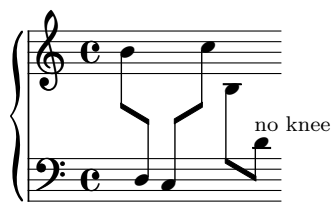
`'beam-concave.ly':`

Concave beams should be horizontal. Informally spoken, concave refers to the shape of the notes that are opposite a beam. If an up-beam has high notes on its center stems, then we call it concave. This example shows borderline cases. Only the beams that are marked ‘horiz’ should be printed horizontally.



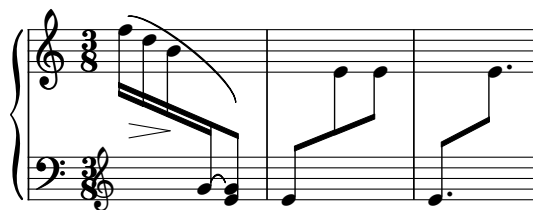
‘beam-cross-staff-auto-knee.ly’:

Automatic cross-staff knees also work (here we see them with explicit staff switches).



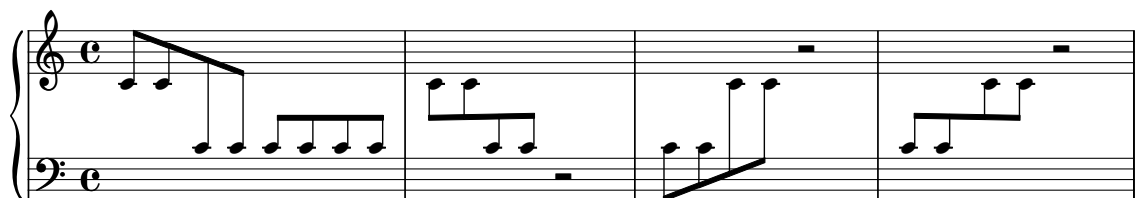
‘beam-cross-staff-slope.ly’:

Cross staff (kneed) beams don’t cause extreme slopes.



‘beam-cross-staff.ly’:

Beams can be typeset over fixed distance aligned staves, beam beautification doesn’t really work, but knees do. Beams should be behave well, wherever the switching point is.



‘beam-damp.ly’:

Beams are less steep than the notes they encompass.



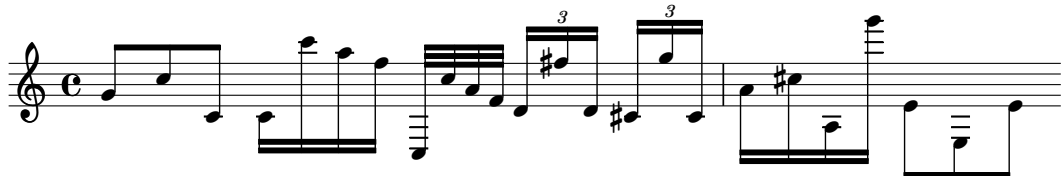
‘beam-default-lengths.ly’:

Beamed stems have standard lengths if possible. Quantization is switched off in this example.



`'beam-extreme.ly':`

Beams should behave reasonably well, even under extreme circumstances. Stems may be short, but noteheads should never touch the beam. Note that under normal circumstances, these beams would get knees here. `Beam.auto-knee-gap` was set to false.



`'beam-french.ly':`

French style beaming. In french beaming, the stems do not go to the outer beams.



`'beam-funky-beamlet.ly':`

Funky kneed beams with beamlets also work. The beamlets should be pointing to the note head.



`'beam-funky.ly':`

Knee beaming, complex configurations. According to Paul Roberts, the first stem of a beam determines the direction of the beam, and as such the way that following (kneaded) stems attach to the beam. This is in disagreement with the current algorithm.



`'beam-knee-symmetry.ly':`

Point-symmetric beams should receive the same quanting. There is no up/down bias in the quanting code.



`'beam-length.ly':`

Beams should look the same.



`'beam-manual-beaming.ly':`

Beaming can be overridden for individual stems.



`'beam-multiple-cross-staff.ly':`

Knead beams (often happens with cross-staff beams) should look good when there are multiple beams: all the beams should go on continuously at the staff change. Stems in both staves reach up to the last beam.



`'beam-over-barline.ly':`

Explicit beams may cross barlines.



`'beam-position.ly':`

Beams on ledgered notes should always reach the middle staff line. The second beam counting from the note head side, should never be lower than the second staff line. This does not hold for grace note beams. Override with `no-stem-extend`.



`'beam-postfix-notation.ly':`

Beams and ties may be entered in postfix notation, separating the notes and the brackets with a dash.



`'beam-quanting-32nd.ly':`

Stem lengths take precedence over beam quants: 'forbidden' quants are only avoided for 32nd beams when they are outside of the staff. However, that leads to very long stems, which is even worse.



`'beam-quanting-horizontal.ly':`

Test beam quant positions for horizontal beams. Staff lines should be covered in all cases. For 32nd beams, the free stem lengths are between 2 and 1.5.



`'beam-quarter.ly':`

Quarter notes may be beamed: the beam is halted momentarily.



`'beam-rest.ly':`

The number of beams doesn't change on a rest.



`'beam-second.ly':`

Seconds are tricky. We used to have problems with seconds being too steep, or getting too long stems. In a file like this, showing seconds, you'll spot something fishy very quickly.



`'beam-shortened-lengths.ly':`

Beams in unnatural direction, have shortened stems, but do not look too short.



`'beamed-chord.ly':`

Hairy case for beam, chord, and automatic knees.



`'beaming-ternary-metrum.ly':`

Automatic beaming works also in ternary time sigs.



`'beaming.ly':`

Beaming is generated automatically. Beams may cross bar lines. In that case, line breaks are forbidden. Yet clef and key signatures are hidden just as with breakable bar lines.



`'beams.ly':`

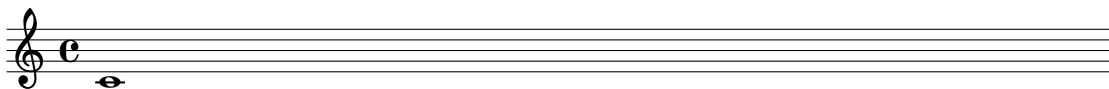
Beams (simple).



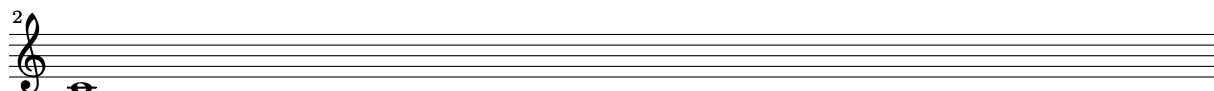
`'between-systems.ly':`

By inserting `\TeX` commands between systems, you can force pagebreaks.

In reality, you'd use the LaTeX command `\newpage` instead of `(pagebreak)` of course.



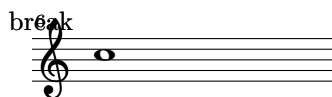
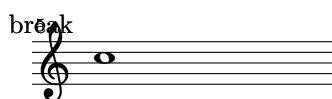
`(pagebreak)`



`'break.ly':`

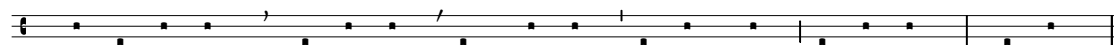
Breaks can be encouraged and discouraged using `\break` and `\noBreak`.





`'breathing-sign-ancient.ly':`

Gregorian chant notation sometimes also uses commas and ticks, but in smaller font size (we call it 'virgula' and 'caesura'). However, the most common breathing signs are *divisio minima*/*maior*/*maxima* and *finalis*, the latter three looking similar to bar glyphs.



`'breathing-sign.ly':`

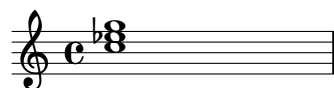
Breathing signs are available in different tastes: commas (default), ticks, vees and 'railroad tracks' (caesura).



`'chord-changes.ly':`

Property `chordChanges`: display chord names only when there's a change in the chords scheme, but always display the chord name after a line break.

Cm



Cm

D

Cm



Cm

D



`'chord-name-entry-11.ly':`

The 11 is only added to natural-3 if it is mentioned explicitly.



‘chord-name-entry.ly’:

Test file for the new chordname entry code (\chords mode): the suffixes are printed below the pitches.

‘chord-name-exceptions.ly’:

The property `chordNameExceptions` can be used to store a list of special notations for specific chords.

Putting the exceptions list encoded as

```
\notes { <c f g bes>1\markup { \super "7" "wahh" } }
```

into `chordNameExceptions` takes a little manoeuvring. The following code transforms `chExceptionMusic` (which is a sequential music) into a list of exceptions.

```
(sequential-music-to-chord-exceptions chExceptionMusic \#t)
```

Then,

```
(append
... ignatzekExceptions)
```

adds the new exceptions to the default ones, which are defined in ‘ly/chord-modifier-init.ly’.

$C^{7/sus4}$

C^{o7}/F

C^{7wahh}

C^{o7}/F

‘chord-name-major7.ly’:

The layout of the major 7 can be tuned with `majorSevenSymbol`.

C^{Δ}
‘chord-names-bass.ly’:

C^{j7}

Test `ignatzek` inversion and bass notes. Above the staff: computed chord names. Below staff: entered chord name.

‘chord-scripts.ly’:

Scripts can also be attached to chord elements.



`'chord-tremolo-short.ly':`

Tremolo repeats can be constructed for short tremolos (total duration smaller than 1/4) too. Only some of the beams are connected to the stems.



`'chord-tremolo.ly':`

Chord tremolos look like beams, but are a kind of repeat symbol. To avoid confusion, chord tremolo beams do not reach the stems, but leave a gap. Chord tremolo beams on half notes are not ambiguous, as half notes cannot appear in a regular beam, and should reach the stems.

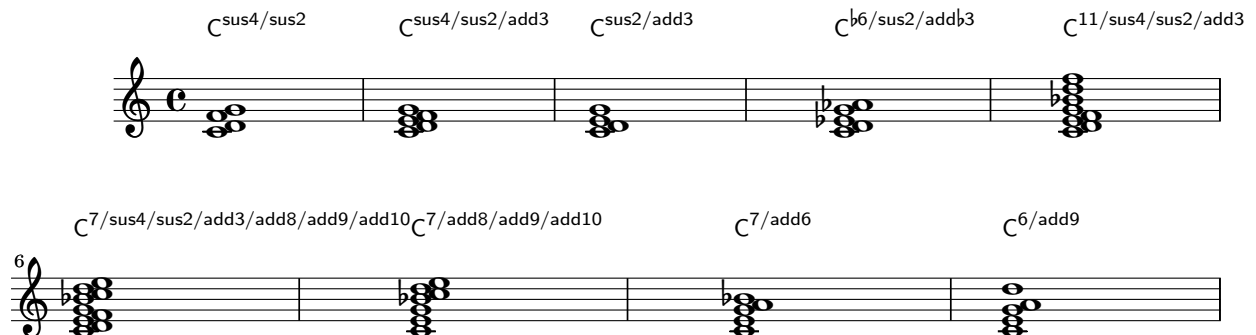
In this example, each tremolo lasts exactly one measure.

(To ensure that the spacing engine is not confused we add some regular notes as well.)



`'chords-funky-ignatzek.ly':`

Jazz chords, unusual combinations.



`'clef-oct.ly':`

Octavation signs may be added to clefs. These octavation signs may be placed below or above (meaning an octave higher or lower), and can take any value, including 15 for two octaves.



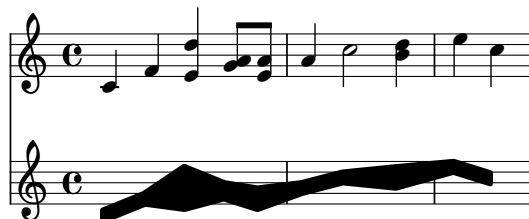
`'clefs.ly':`

Clefs with `full-size-change` should be typeset in full size. For octaviated clefs, the “8” should appear closely above or below the clef respectively.



‘cluster.ly’:

Clusters are a device to denote that a complete range of notes is to be played.



‘collision-2.ly’:

Collisions for single head notes.



‘collision-dots.ly’:

Collision resolution tries to put notes with dots on the right side.



‘collision-head-chords.ly’:

Note heads in collisions should be merged if they have the same positions in the extreme note heads.



‘collision-heads.ly’:

If `merge-differently-headed`, then open note heads may be merged with black noteheads, but only if the black note heads are from 8th or shorter notes.



‘collision-merge-differently-dotted.ly’:

If `NoteCollision` has `merge-differently-dotted` set, note heads that have differing dot counts may be merged anyway.



`'collision-mesh.ly':`

Oppositely stemmed chords, meshing into each other, are resolved.



`'collisions.ly':`

Normal collisions. We have support for polyphony, where the middle voices are horizontally shifted.



`'completion-heads-polyphony.ly':`

Completion heads are broken across bar lines. This was intended as a debugging tool, but it can be used to ease music entry. Completion heads are not fooled by polyphony with a different rhythm.



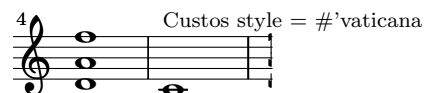
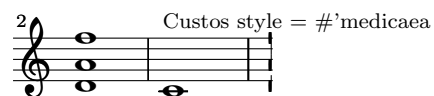
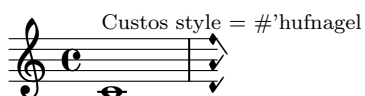
`'completion-heads.ly':`

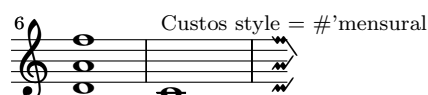
If the `Note_heads_engraver` is replaced by the `Completion_heads_engraver`, notes that cross bar lines are split into tied notes.



`'custos.ly':`

Custodes in various styles.





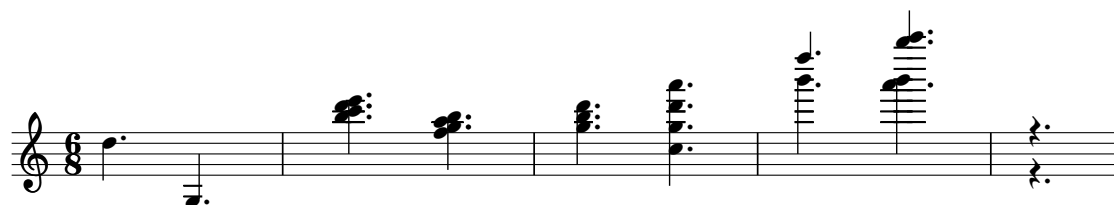
'dot-flag-collision.ly':

Dots move to the right when a collision with the (up)flag happens.



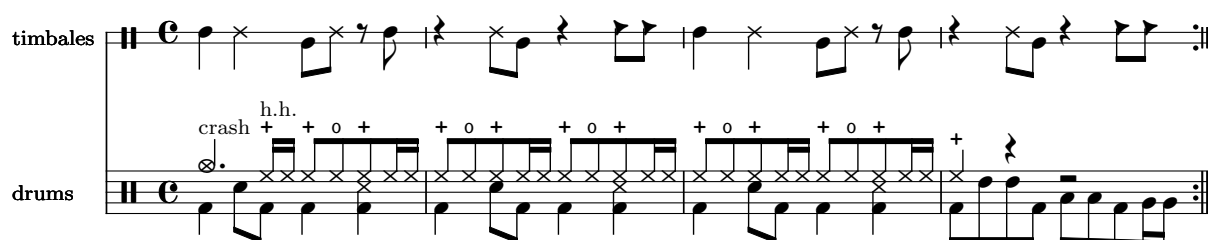
'dots.ly':

Noteheads can have dots, and rests can too. Augmentation dots should never be printed on a staff line, but rather be shifted vertically. They should go up, but in case of multiple parts, the down stems have down shifted dots. (Wanske p. 186) In case of chords, all dots should be in a column. The dots go along as rests are shifted to avoid collisions.



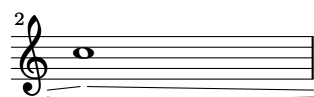
'drums.ly':

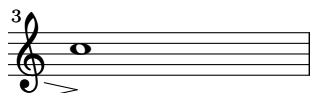
Drum notation, although kludgy, should work.



'dynamics-broken-hairpin.ly':

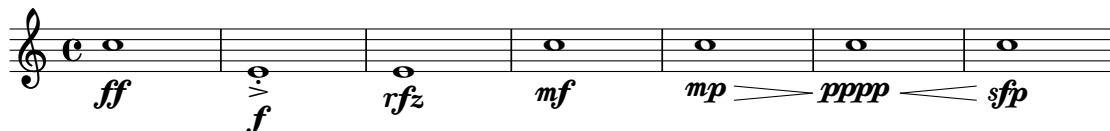
Broken crescendi should be open on one side.





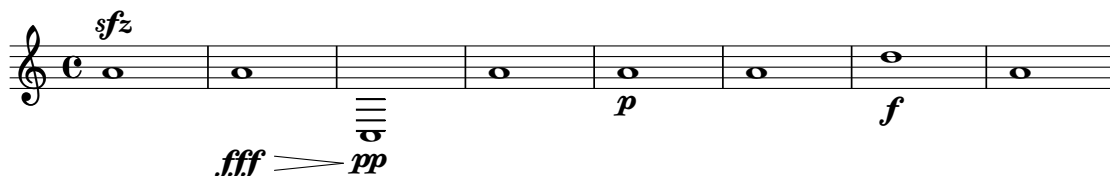
`'dynamics-glyphs.ly':`

Dynamic letters are kerned, and their weight matches that of the hairpin signs. The dynamic scripts should be horizontally centered on the note head. Scripts that should appear closer to the note head (staccato, accent) are reckoned with.



`'dynamics-line.ly':`

Dynamics appear below or above the staff. If multiple dynamics are linked with (de)crescendi, they should be on the same line. Isolated dynamics may be forced up or down.



`'dynamics-unbound-hairpin.ly':`

Crescendi may start off-notes. In that case, they should not collapse into flat lines.



`'easy-notation.ly':`

Ez-notation prints names in note heads. You also get ledger lines, of course.



`'figured-bass.ly':`

Figured bass is created by the FiguredBass context which eats figured bass requests and rest-requests. You must enter these using the special `\figures { }` mode, which allows you to type numbers, like `<<4 6+>>`.

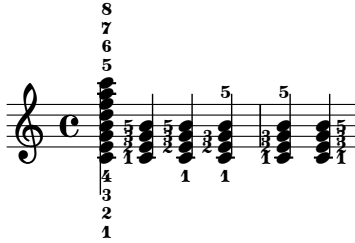
You can also type letters by entering quoted strings, as demonstrated in the last example.

$$\begin{array}{ccccccc} & & \begin{bmatrix} 11 \\ 9 \end{bmatrix} & & 7 & 7 & \\ & & 7 & 7 & 5 & 5 & \begin{bmatrix} 6 \\ \text{bla} \end{bmatrix} \\ \begin{bmatrix} 7 \\ 5 \end{bmatrix} & & \begin{bmatrix} 5 \\ 3 \end{bmatrix} & \begin{bmatrix} 5 \\ 3 \end{bmatrix} & \begin{bmatrix} 5 \\ 3 \end{bmatrix} & \begin{bmatrix} 5 \\ 3 \end{bmatrix} & \begin{bmatrix} 5 \\ 7 \end{bmatrix} \\ 3 & 3 & 3 & 3\# & 3 & 3 & \text{v7} \end{array}$$



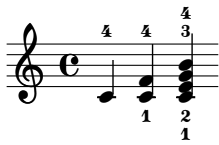
'finger-chords.ly':

With the new chord syntax it's possible to associate fingerings uniquely with notes. This makes horizontal fingering much easier to process.



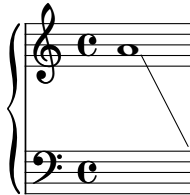
'fingering.ly':

Automatic fingering tries to put fingering instructions next to noteheads.



'follow-voice-break.ly':

When put across line breaks, only the part before the line break is printed. The line-spanners connects to the Y position of the note on the next line.



'font-magnification.ly':


The magnification can be set for any font. Note that this doesn't change variable symbols such as beams or slurs.



`'font-name.ly':`

Using other fonts can be done by setting font-name for the appropriate object. This may include Postscript fonts that are available through (La)TeX.

Wait for Utopia Italic



This text is Dunhill

`'generic-output-property.ly':`

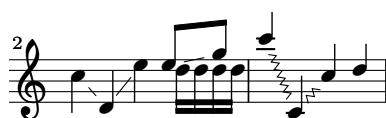
As a last resort, the placement of grobs can be adjusted manually, by setting the **extra-offset** of a grob. A



`'glissando.ly':`

Simple glissando lines between notes are supported. The first two glissandi are not consecutive.

The engraver does no time-keeping, so it involves some trickery to get `<< { s8 s8 s4 } { c4 \gliss d4 } >>` working correctly.



`'grace-auto-beam.ly':`

The autobeamer is not confused by grace notes.



`'grace-bar-line.ly':`

Bar line should come before the grace note.



`'grace-bar-number.ly':`

Grace notes do tricky things with timing. If a measure starts with a grace note, the measure does not start at 0, but earlier. Nevertheless, lily should not get confused. For example, line breaks should be possible at grace notes, and the bar number should be printed correctly.





'grace-beam.ly':

Grace beams and normal beams may occur simultaneously. Unbeamed grace notes are not put into normal beams.



'grace-end.ly':

Grace notes after the last note do not confuse the timing code.



'grace-nest.ly':

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



'grace-nest1.ly':

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



'grace-nest2.1y':

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



'grace-nest3.ly':

Another nested grace situation.



'grace-nest4.ly':

Another combination of grace note nesting.



`'grace-nest5.ly':`

Another nested grace situation.



`'grace-part-combine.ly':`

Partcombiner and grace notes can go together.



`'grace-staff-length.ly':`

Stripped version of trip.ly. Staves should be of correct length.



`'grace-start.ly':`

Pieces may begin with grace notes.



`'grace-stems.ly':`

Here startGraceMusic should set no-stem-extend to true; the two grace beams should be the same here.



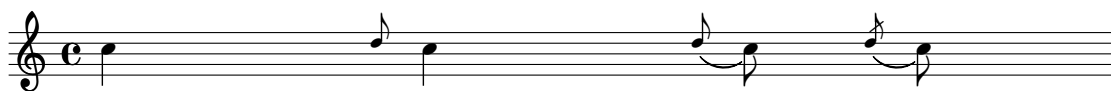
`'grace-sync.ly':`

Grace notes in different voices/staves are synchronized.



`'grace-types.ly':`

Different grace types explained: the base grace switches to smaller type. The appoggiatura also inserts a slur, and the acciaccatura inserts a slur and slashes the stem.



`'grace-unfold-repeat.ly':`

Grace notes and unfolded repeats. Line breaks may happen before grace notes.



`'grace-volta-repeat-2.ly':`

Graces at combined with volta repeats: a repeat starting with a grace, following a repeat directly. The bars should be merged into one `:| |:`.



`'grace-volta-repeat.ly':`

Repeated music can start with grace notes. Bar checks preceding the grace notes do not cause synchronization effects.



`'grace.ly':`

Grace notes are typeset as an encapsulated piece of music. You can have beams, notes, chords, stems etc. within a `\grace` section. Slurs that start within a grace section, but aren't ended are attached to the next normal note. Grace notes have zero duration. If there are tuplets, the grace notes won't be under the brace. Grace notes can have accidentals, but they

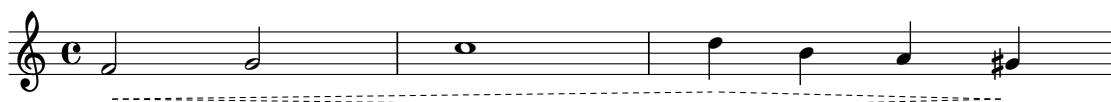
are (currently) spaced at a fixed distance. Grace notes (of course) come before the accidentals of the main note. Grace notes can also be positioned after the main note.

Grace notes without beams should have a slash, if `flagStyle` is not set. Main note scripts don't end up on the grace note.



`'hairpin-dashed.ly':`

Hairpin crescendi may be dashed.



`'hairpin-ending.ly':`

Hairpin dynamics start under notes if there are no text-dynamics. If there are text dynamics, the hairpin does not run into them.



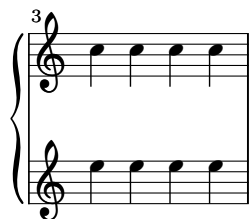
`'hara-kiri-pianostaff.ly':`

Hara kiri staves kill themselves if they are empty. This example really contains three staves, but as they progress, empty ones are removed: this example has three staves, but some of them disappear: note how the 2nd line only has the bar number 2. (That the bar number is printed might be considered a bug, however, the scenario of all staves disappearing doesn't happen in practice.)

Any staff brackets and braces are removed, both in the single staff and no staff case.

This example was done with a pianostaff, which has fixed distance alignment; this should not confuse the mechanism.





`'instrument-name-markup.ly':`

Instrument names are set with `Staff.instrument` and `Staff.instr`. You can enter markup texts to create more funky names, including alterations.



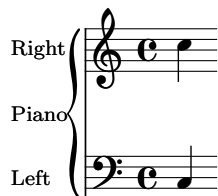
`'instrument-name-partial.ly':`

Instrument names are also printed on partial starting measures.



`'instrument-name.ly':`

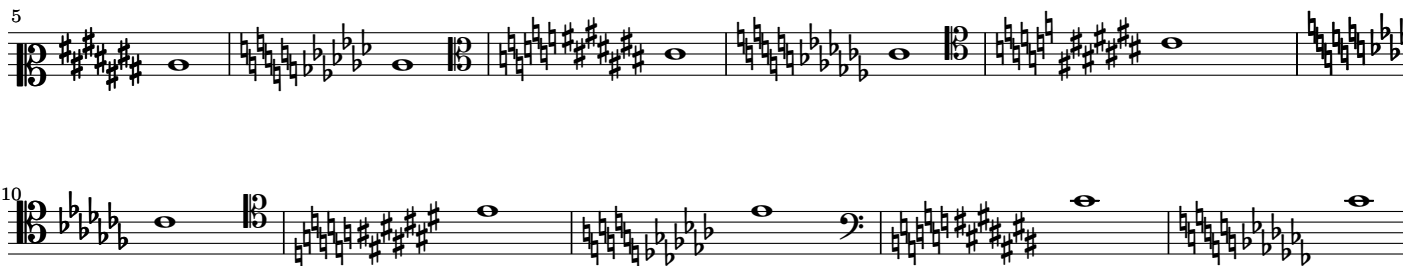
Staff margins are also markings attached to barlines. They should be left of the staff, and be centered vertically wrt the staff. They may be on normal staves, but also on compound staves, like the PianoStaff.



`'key-clefs.ly':`

Tests placement of accidentals in every clef.





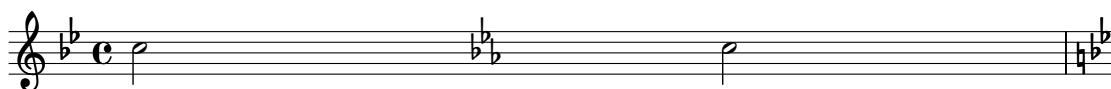
‘key-signature-scordatura.ly’:

Key signatures can be set per pitch individually. This can be done by setting `Staff.keySignature` directly.



‘keys.ly’:

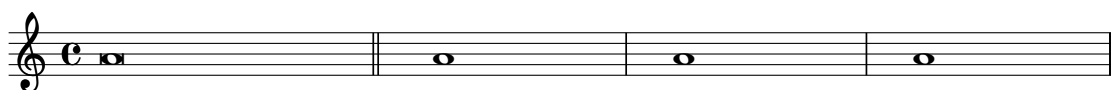
Key signatures appear on key changes. They may also appear without barlines. The restoration accidentals are not printed at the start of the line. If `createKeyOnClefChange` is set, they’re also created on a clef change.



‘lyric-align.ly’:

Lyric alignment is adjustable both in terms of alignment between stanzas and on notehead.

If the property alignment is not set, there is automatic determination of alignment type based on punctuation (, see `lyric-phrasing.ly`).



Particularly useful for reciting notes left centered right
with many syllables under them. l c r



left half way left one quarter left one tenth left one whole
l l l x



Very short lyrics remain in touch with their note

Unless ignore-length-mismatch is true

x

x

`'lyric-combine-polyphonic.ly':`

Polyphonic rhythms and rests don't disturb `\addlyrics`.



Do mi nus ex

Do na

`'lyric-combine.ly':`

Lyrics can be set to a melody automatically. Excess lyrics will be discarded. Lyrics will not be set over rests. You can have melismata either by setting a property `melismaBusy`, or by setting `automaticMelismas` (which will set melismas during slurs and ties). If you want a different order than first Music, then Lyrics, you must precook a chord of staves/lyrics and label those. Of course `\rhythm` ignores any other rhythms in the piece. Hyphens and extenders do not assume anything about lyric lengths, so they continue to work.

la — la — la la la la

da — da — da-da da da

melisma

`'lyric-extender.ly':`

Tests lyric extenders.

bla alb xxx yyy

`'lyric-hyphen.ly':`

Tests lyric hyphens.

bla — alb xxx — yyy

‘lyric-phrasing.ly’:

Lyric phrasing

We find start and end of phrases, and align lyrics of multiple stanzas accordingly.

Also, lyrics at start of melismata should be left aligned. (is that only lyrics that are followed by ‘_’? Because that seems to be the case now – jcn)

| | | | |
x| x| x| x| x|

1: Start sentence melisma end.
2: x x x_____ x

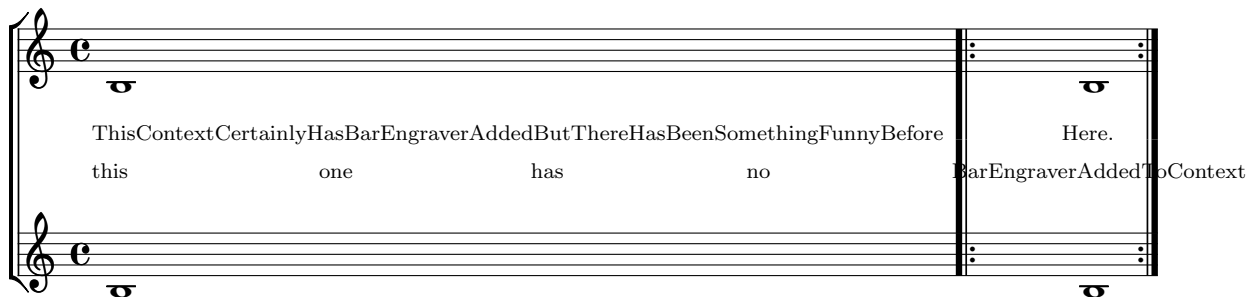
Only lyrics that are followed by ‘_’ while there’s a melisma, are left-aligned, in this case the third x.



1: Start sentence melisma end.
2: x x x_____ x.

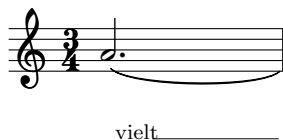
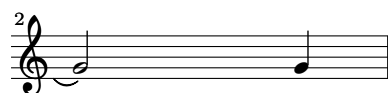
‘lyrics-bar.ly’:

Adding a Bar_engraver to the LyricsVoice context makes sure that lyrics don’t collide with barlines.

A musical score with two staves, treble and bass clef, in common time. The lyrics are: 'ThisContextCertainlyHasBarEngraverAddedButThereHasBeenSomethingFunnyBefore' on the first line, 'this one has no' on the second line, and 'Here.' on the third line. A thick vertical bar line is placed after the word 'no'. The text 'BarEngraverAddedToContext' is written to the right of the bar line. The lyrics are aligned with the bar lines.

‘lyrics-extender.ly’:

Extenders that end a staff should not extend past the staff. Also shown: a trick to get an extender at the end of the staff.

A musical staff in treble clef with a 3/4 time signature. It contains a melody of a quarter note C4 followed by a half note D4. The half note D4 is a melisma, represented by a long horizontal line extending to the right. The lyrics 'vielt' are written below the staff.A musical staff in treble clef with a 3/4 time signature. It contains a melody of a quarter note C4 followed by a half note D4. The half note D4 is a melisma, represented by a long horizontal line extending to the right. The lyrics 'Zeit' are written below the staff.

‘lyrics-melisma-beam.ly’:

Melismata are triggered by manual beams.



bla bla bla

`'lyrics-multi-stanza.ly':`

Lyrics syllables are aligned according to punctuation. Stanza and stz set stanza numbers.

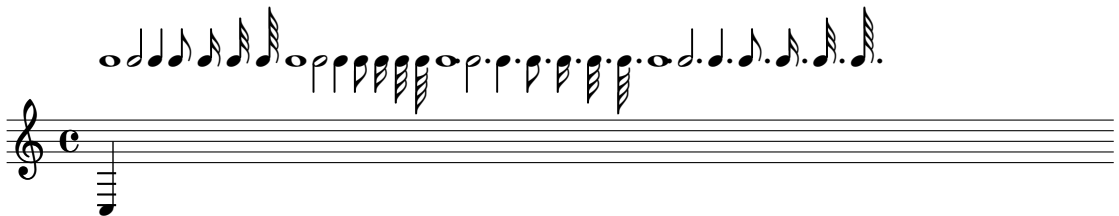


Bert Hi, my name is bert.

Ernie Ooooo, ché — ri, je t'aime.

`'markup-note.ly':`

The note markup function is used to make metronome markings. It works for a variety of flag dot and duration settings.



`'markup-stack.ly':`

Stacking of markup scripts.



`'measure-grouping.ly':`

The Measure_grouping_engraver adds triangles and brackets above beats when you set beat-Grouping. This shows a fragment of Steve Martland's Dance Works.



`'mensural.ly':`

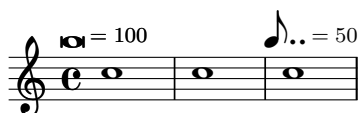
There is limited support for mensural notation: note head shapes are available. Mensural stems are centered on the note heads, both for up and down stems.



`'metronome-marking.ly':`

Here `empo` directives are printed as metronome markings.

The marking is left aligned with the time signature, if there is one.



`'mm-rests2.ly':`

If `Score.skipBars` is set, the signs for four, two, and one measure rest are combined to produce the graphical representation of rests for up to 10 bars. The number of bars will be written above the sign.



`'mmrest-collision.ly':`

Tests a collision between multimeasure rests in different voices.



`'molecule-hacking.ly':`

You can write molecule callbacks in Scheme, thus providing custom glyphs for notation elements. A simple example is adding parentheses to existing molecule callbacks.

The parenthesized beam is less successful due to implementation of the Beam. The note head is also rather naive, since the extent of the parens are also not seen by accidentals.



`'multi-measure-rest-center.ly':`

The multimeasure rest is centered exactly between bar lines.



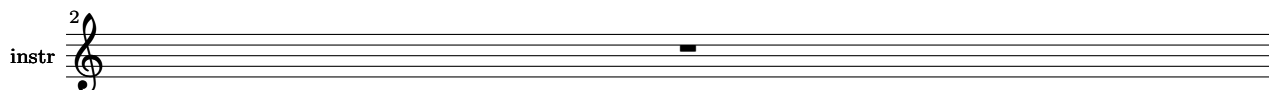
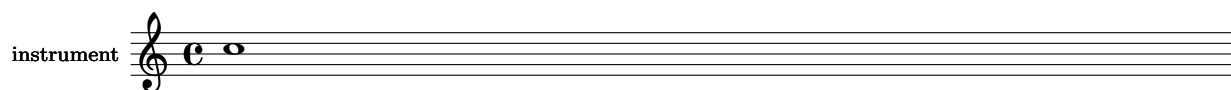
`'multi-measure-rest-grace.ly':`

Grace notes and multi-measure rests.



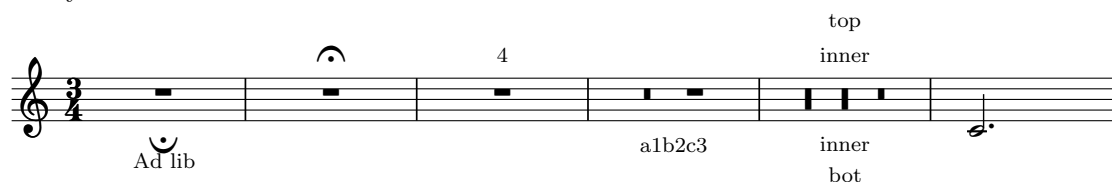
`'multi-measure-rest-instr-name.ly':`

This combines instrument names and multimeasure rests (there was an interesting bug in 1.3.98).



`'multi-measure-rest-text.ly':`

Texts may be added to the multi measure rests.



`'multi-measure-rest.ly':`

Multiple measure rests do not collide with barlines and clefs. They are not expanded when you set `Score.skipBars`. Although the multi-measure-rest is a `Spanner`, minimum distances are set to keep it colliding from barlines.

Rests over measures during longer than 2 wholes use breve rests. When more than 10 or more measures (tunable through `expand-limit`) are used then a different symbol is used.



`'music-map.ly':`

With `music-map`, you can apply functions operating on a single piece of music to an entire music expression. In this example, the the function `notes-to-skip` changes a note to a skip. When applied to an entire music expression in the 1st measure, the scripts and dynamics are left over. These are put onto the 2nd measure.



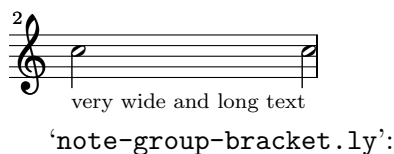
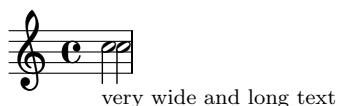
`'new-markup-syntax.ly':`

New markup syntax.

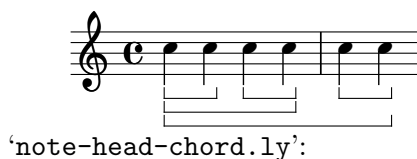


`'non-empty-text.ly':`

Text is set with empty horizontal dimensions. The boolean property `no-spacing-rods` in `TextScript` is used to control the horizontal size of text.



Note grouping events are used to indicate where brackets for analysis start and end.

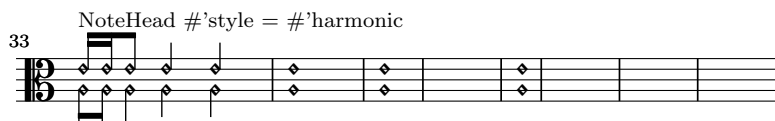
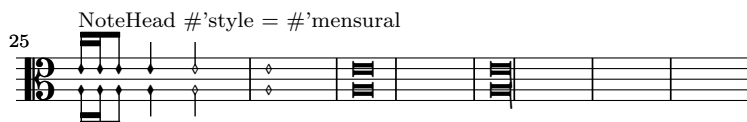
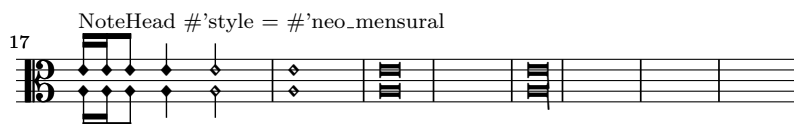
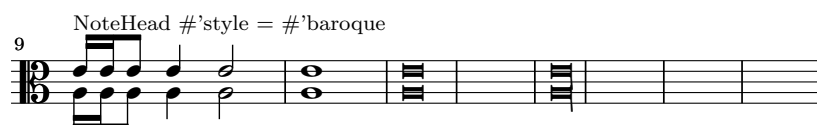
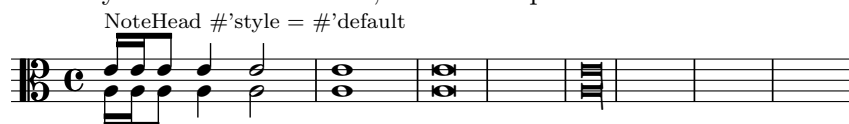


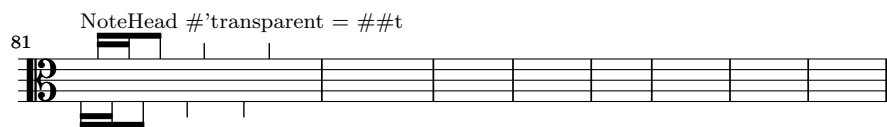
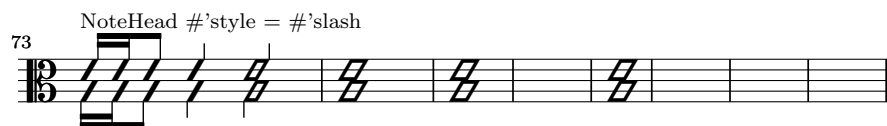
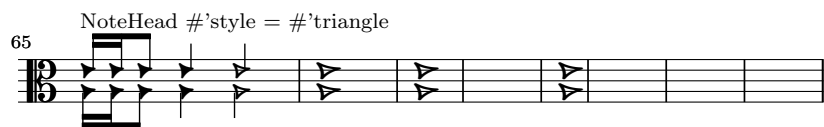
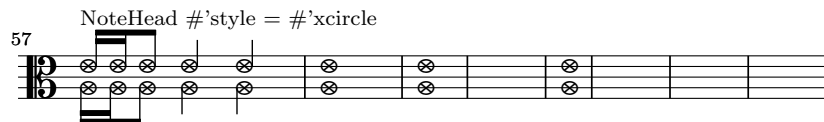
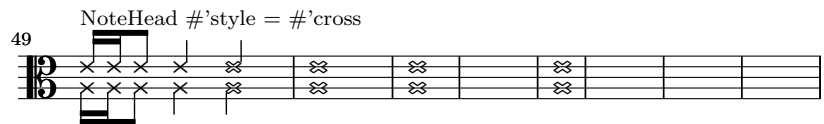
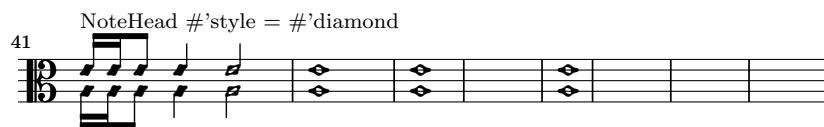
Note heads are flipped on the stem to prevent collisions. It also works for whole heads that have invisible stems.



Note head shapes are settable. The stem endings should be adjusted per note head. If you want different note head styles on one stem, you must create a special context called Thread.

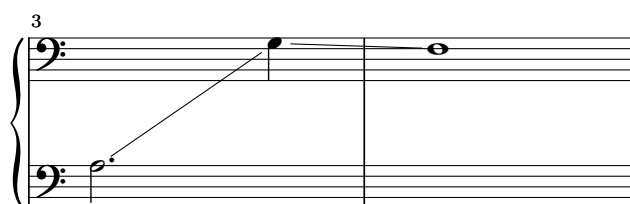
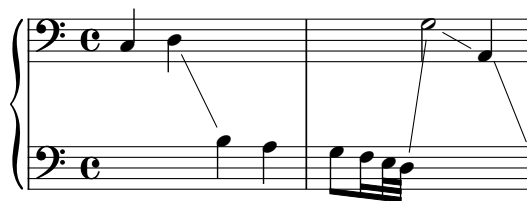
Harmonic notes have a different shape and different dimensions. Nevertheless, noteheads in both styles can be combined, on either up or down stems.





`'note-line.ly':`

Note head lines (e.g. glissando) run between centers of the note heads.



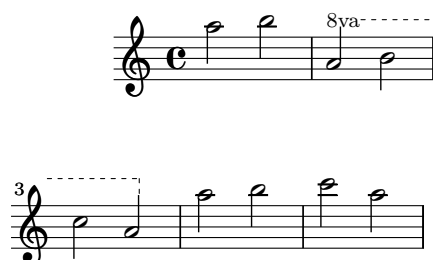
`'number-staff-lines.ly':`

The number of stafflines of a staff can be set. Ledger lines both on note heads and rests are adjusted. Barlines also are adjusted.



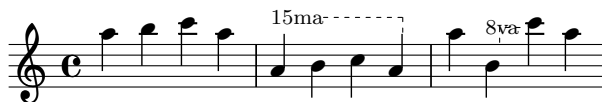
`'ottava-broken.ly':`

Ottava brackets behave properly at line breaks: no vertical line, and the horizontal line doesn't stick out.



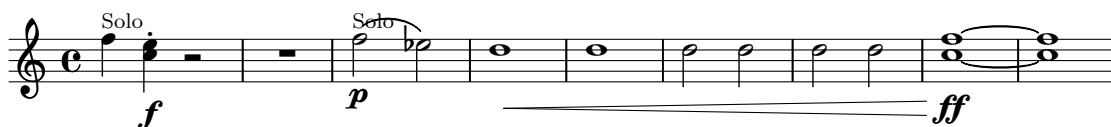
`'ottava.ly':`

Ottava brackets are supported, through the use of the scheme function `set-octavation`.



`'pc-mmrest.ly':`

Multi measure rests of second voice should not disappear.



`'pc-switch-slur.ly':`

The partcombiner should not combine two small slurs into a big one.



`'phrasing-slur.ly':`

Slurs play well with phrasing slur.



`'prefatory-empty-spacing.ly':`

The A is atop an invisible barline. The barline although invisible, is also translated because it is the last one of the break alignment.



`'prefatory-spacing-matter.ly':`

Prefatory spacing.

TODO: Show all common combinations to check for spacing anomalies.



`'property-once.ly':`

Once properties take effect during a single time step only.



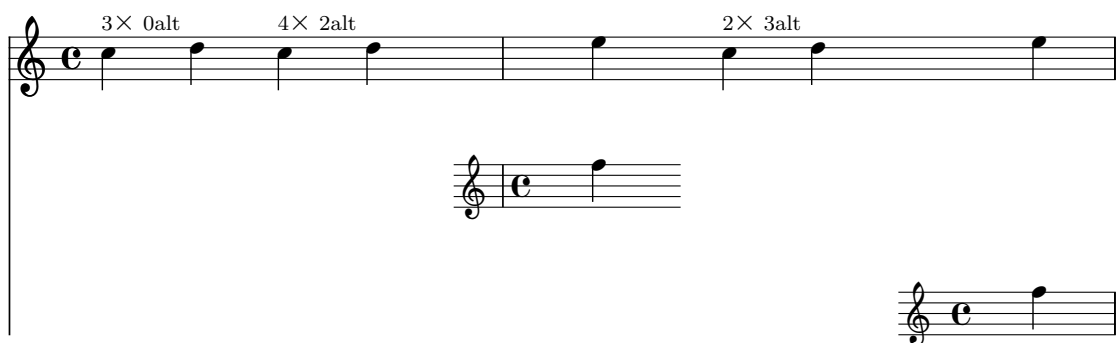
`'rehearsal-mark.ly':`

Rehearsal marks are printed over barlines. They can be incremented automatically or manually.



`'repeat-fold.ly':`

Folded. This doesn't make sense without alternatives, but it works.



`'repeat-line-break.ly':`

Across linebreaks, the left edge of a first and second alternative bracket should be equal.



`'repeat-percent-skipbars.ly':`

Percent repeats are not skipped, even when skipBars is set.



`'repeat-percent.ly':`

Measure repeats are supported, and may be nested with beat repeats.



`'repeat-slash.ly':`

Beat repeats are supported.



`'repeat-unfold-all.ly':`

Repeats may be unfolded through the Scheme function `unfold-repeats`.



`'repeat-unfold.ly':`

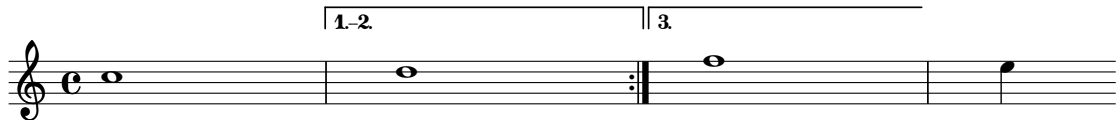
LilyPond has three modes for repeats: folded, unfolded and semi-unfolded. Unfolded repeats are fully written out. Semi unfolded repeats have the body written and all alternatives sequentially. Folded repeats have the body written and all alternatives simultaneously. If the number of alternatives is larger than the repeat count, the excess alternatives are ignored. If the number of alternatives is smaller, the first alternative is multiplied to get to the number of repeats.

Unfolded behavior:



`'repeat-volta-skip-alternatives.ly':`

When too few alternatives are present, the first alternative is repeated, by printing a range for the 1st repeat.



`'repeat-volta.ly':`

Volta (Semi folded) behavior. Voltas can start on non-barline moments. If they don't barlines should still be shown.



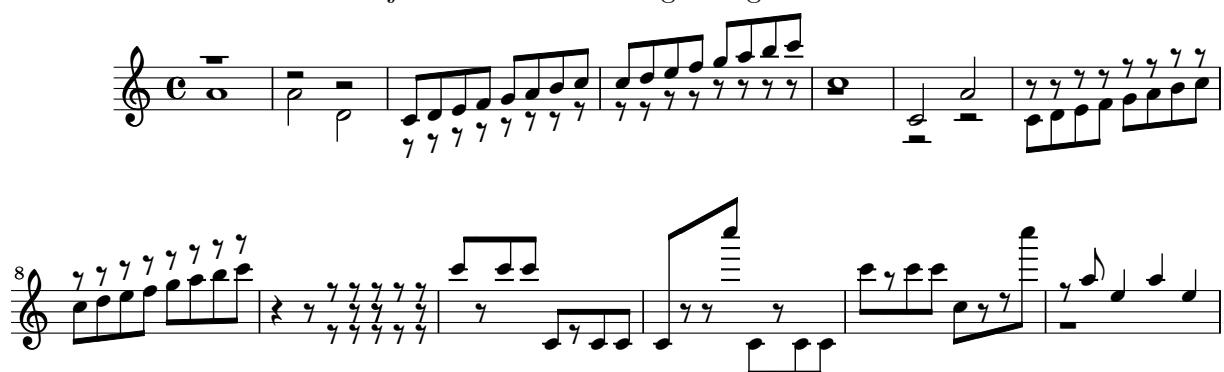
`'rest-collision-default.ly':`

Rests in collisions sit opposite of the note if no direction is specified for the voice containing the rest.



`'rest-collision.ly':`

Rests should not collide with beams, stems and noteheads. Rests may be under beams. Rests should be move by integral number of spaces inside the staff, and by half spaces outside. Notice that the half and whole rests just outside the staff get ledger lines in different cases.



`'rest-ledger.ly':`

Whole and half rests moving outside the staff should get ledger lines.



`'rest-pitch.ly':`

Rests can have pitches—these will be affected by transposition and relativization. If a rest has a pitch, rest collision will leave it alone.



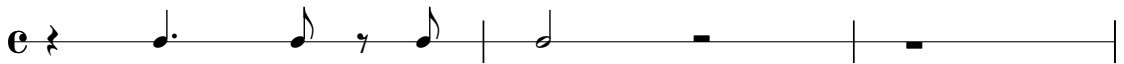
`'rest.ly':`

Rests. Note that the dot of 8th, 16th and 32nd rests rest should be next to the top of the rest. All rests except the whole rest are centered on the middle staff line.



`'rhythmic-staff.ly':`

In rhythmic staves, stems should go up, and bar lines have the size for a 5 line staff. The whole note hangs from the rhythmic staff.



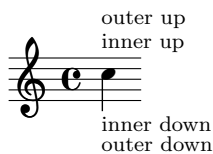
`'script-collision.ly':`

Scripts are put on the utmost head, so they are positioned correctly when there are collisions.



`'script-stack-order.ly':`

Scripts can be stacked. The order is determined by a priority field, but when objects have the same priority, the input order determines the order. Objects specified first are closest to the note.



`'script-stacked.ly':`

Scripts may be stacked.



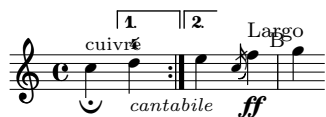
`'size11.ly':`

Different text styles are used for various purposes.



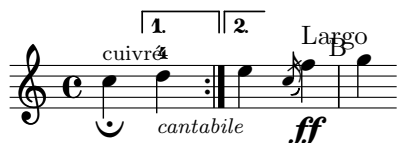
‘size13.ly’:

Different text styles are used for various purposes.



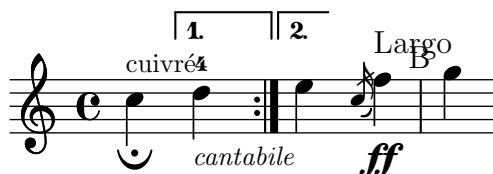
‘size16.ly’:

Different text styles are used for various purposes.



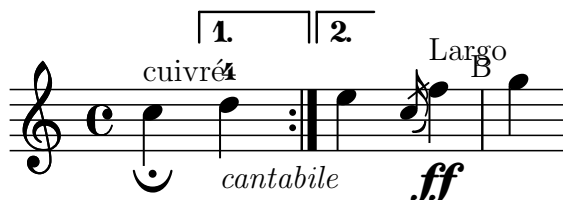
‘size20.ly’:

Different text styles are used for various purposes.



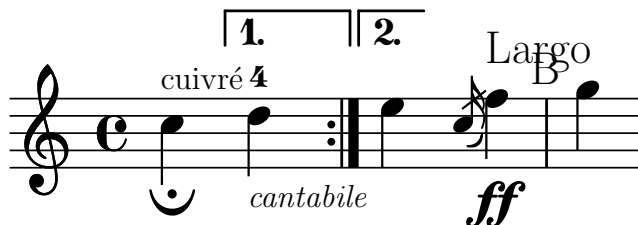
‘size23.ly’:

Different text styles are used for various purposes.



‘size26.ly’:

Different text styles are used for various purposes.



‘slur-area.ly’:

The area underneath an (up) slur is minimised to improve the shape.





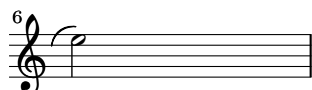
`'slur-attachment.ly':`

Slurs should be attached to note heads, except when they would collide with beams.



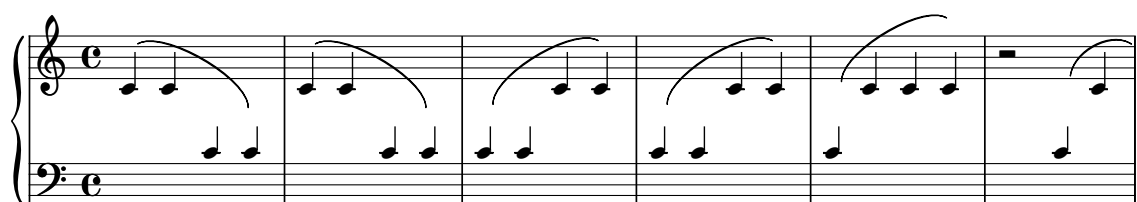
`'slur-broken-trend.ly':`

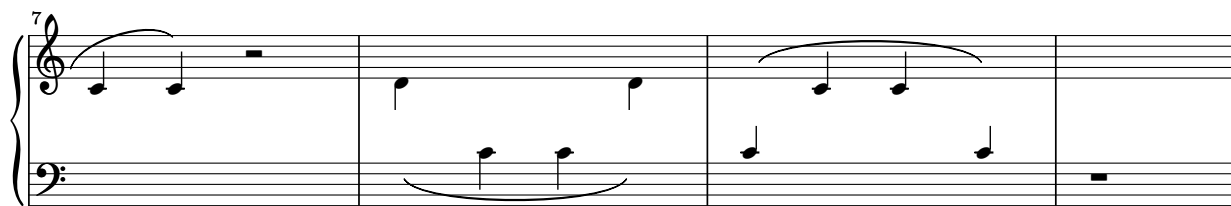
Across line breaks, slurs behave nicely. On the left, they extend to just after the preferatory matter, and on the right to the end of the staff. A slur should follow the same vertical direction it would have in unbroken state.



`'slur-cross-staff.ly':`

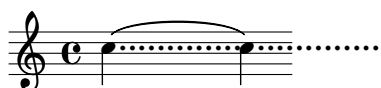
The same goes for slurs. They behave decently when broken across linebreak.





`'slur-dots.ly':`

Slurs should not get confused by augmentation dots. We use a lot of dots here, to make problems more visible.



`'slur-nice.ly':`

Slurs should look nice and symmetric. The curvature may increase only to avoid noteheads, and as little as possible. Slurs never run through noteheads or stems.



`'slur-rest.ly':`

Slurs may be placed over rest. The slur will avoid colliding with the rest.



`'slur-staccato.ly':`

Manual hack for slur and staccato.



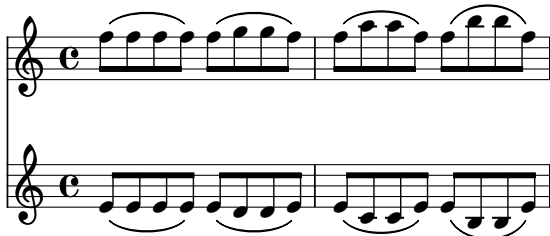
`'slur-stem-broken.ly':`

Trend of broken slur with user-overridden stem attachment should also follow the same vertical direction it would have had in unbroken state.



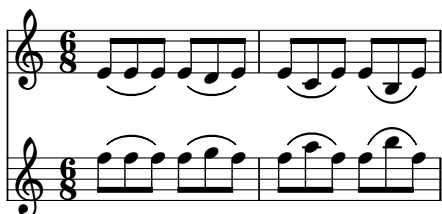
`'slur-symmetry-1.ly':`

Symmetric figures should lead to symmetric slurs.



`'slur-symmetry.ly':`

Symmetric figures should lead to symmetric slurs.



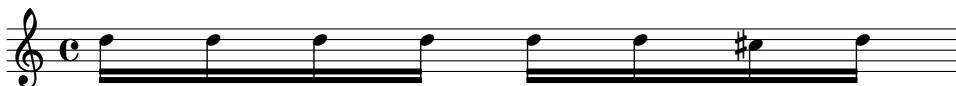
`'spacing-accidental-staffs.ly':`

Accidentals in different staves don't effect the spacing of the quarter notes here.



`'spacing-accidental-stretch.ly':`

Accidentals don't influence the amount of stretchable space.



`'spacing-accidental.ly':`

Accidentals sticking out to the left of a note will take a little more space, but only if the spacing is tight.



`'spacing-bar-stem.ly':`

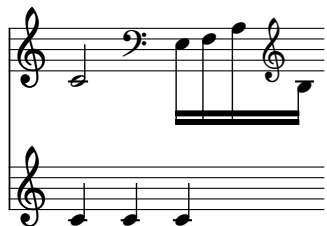
Downstem notes following a barline are printed with some extra space. This is an optical correction similar to juxtaposed stems.

Accidentals after the barline get some space as well.



`'spacing-clef-first-note.ly':`

Clef changes at the start of a line get much more space than clef changes halfway the line.



`'spacing-end-of-line.ly':`

Broken matter at the end of line does not upset the space following rests and notes.



`'spacing-ended-voice.ly':`

A voicelet (a very short voice to get polyphonic chords correct) should not confuse the spacing engine.



`'spacing-folded-clef.ly':`

A clef can be folded below notes in a different staff, if this doesn't disrupt the flow of the notes.



`'spacing-folded-clef2.ly':`

A clef can be folded below notes in a different staff, if there is space enough. With `Paper_column` molecule callbacks we can show where columns are in the score.



`'spacing-grace-duration.ly':`

Spacing uses the duration of the notes, but disregards grace notes for this. In this example, the 8ths around the grace are spaced exactly as the other 8th notes.



`'spacing-grace.ly':`

Grace note spacing. Should be tuned?



`'spacing-knee.ly':`

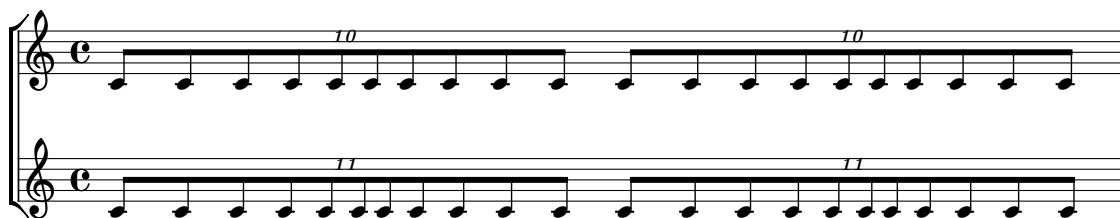
For knees, the spacing correction is such that the stems are put at regular distances. This effect takes into account the width of the note heads and the thickness of the stem.



`'spacing-multi-tuplet.ly':`

Concurrent tuplets should be spaced equidistantly on all staves.

Note that this only spaces correctly (exactly) when `raggedright` is. For `non-raggedright`, it still shows a bug: uneven spacing.



`'spacing-note-flags.ly':`

The flags of 8th notes take some space, but not too much: the space following a flag is less than the space following a beamed 8th head.



`'spacing-rest.ly':`

Rests get a little less space, since they are narrower. However, the feta quarter rest is relatively wide, causing this effect to be very small.



`'spacing-short-notes.ly':`

Notes that are shorter than the common shortest note, Get a space (i.e. without the space needed for the note) proportional to their duration. So 16th notes get 1/2 of the space of an eighth note. The total distance for a 16th is (including note head) is 3/4 of the eighth note.



`'spacing-stem-bar.ly':`

Upstem notes before a barline are printed with some extra space. This is an optical correction similar to juxtaposed stems.



`'spacing-stem-direction.ly':`

LilyPond corrects for optical spacing of stems. The overlap between to adjacent stems of different direction is used as a measure for how much to correct.



`'spacing-stem-same-direction.ly':`

For juxtaposed chords with the same direction, a slight optical correction is used. It is constant, and only works if two chords have no common head-positions range.



`'spacing-to-grace.ly':`

Space from a normal note/barline to a grace note is smaller than to a normal note.



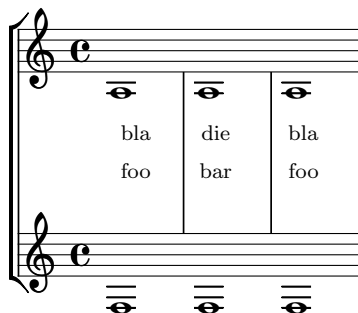
`'spacing-very-tight.ly':`

When tightly spaced, hinterfleisch -> 0. Stems may touch the bar lines, opposite stems may touch eachother. We need a minimum of about a note-width/interline space in these situations, so that in tightly spaced music all vertical lines are about equally spaced.



`‘span-bar.ly’:`

Span bars draw only in between staff bar lines, so setting those to transparent shows bar lines between systems only.



`‘staccato-pos.ly’:`

The staccato dot (and all scripts with follow-into-staff set), must not be on staff lines.



`‘staff-tweak.ly’:`

The staff is a grob, and may be adjusted as well: this one shows a staff with 6 thick line, and a slightly large staffspace. Beams remain correctly quantized.



`‘stanza-number.ly’:`

Stanza numbers may differ for the first and following systems.

first Foo



32nd Bar



`‘stem-direction.ly’:`

Beams, stems and noteheads often have communication troubles, since the two systems for y dimensions (1 unit = staffspace, 1 unit = 1 point) are mixed.

Stems in forced directions (as well as the ones starting from the middle line) are shortened.

‘stem-spacing.ly’:

Stem tremolos or rolls are tremolo signs that look like beam segments crossing stems. If the stem is in a beam, the tremolo must be parallel to the beam. If the stem is invisible (e.g. on a whole note), the tremolo must be centered on the note.

The first system of the musical score for 'The Rose Tree' consists of two staves. The top staff is in treble clef with a common time signature (C). It contains a sequence of notes: a whole note C, a half note G, a quarter note E, a quarter note D, a half note C, a quarter note G, an eighth note E, and an eighth note D. The bottom staff is in treble clef and contains a sequence of notes: a quarter note G, a quarter note E, a quarter note D, a quarter note C, a quarter note G, a quarter note E, a quarter note D, and a quarter note C.

LilyPond correctly determines the size of every system. This includes postscript constructs such as slurs.

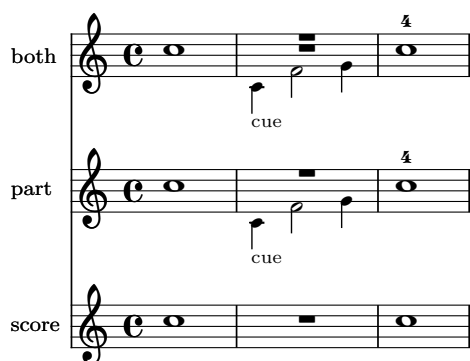
'system-start-bracket.ly':

The piano brace should be shifted horizontally if it is enclosed in a bracket.



`'tag-filter.ly':`

The `\tag` command marks music expressions with a name. These tagged expressions can be filtered out later. This mechanism can be used to make different versions of the same music. In this example, the top staff displays the music expression with all tags included. The bottom two staves are filtered: the part has cue notes and fingerings, but the score has not.



`'text-spanner.ly':`

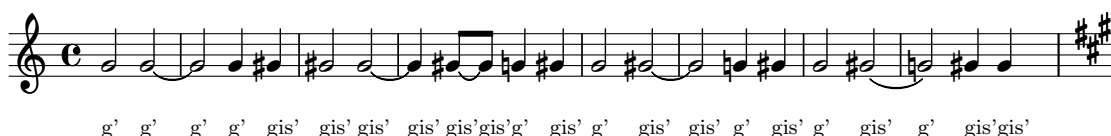
Text spanners should not repeat start text when broken.

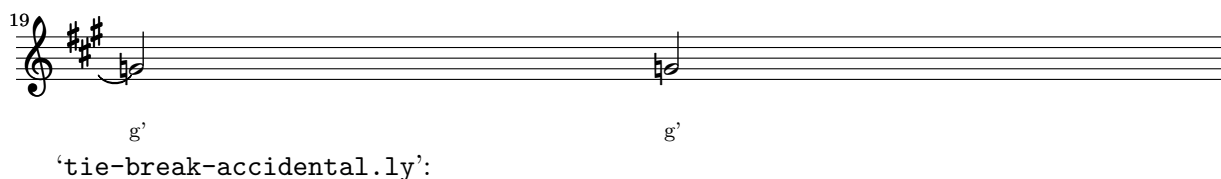
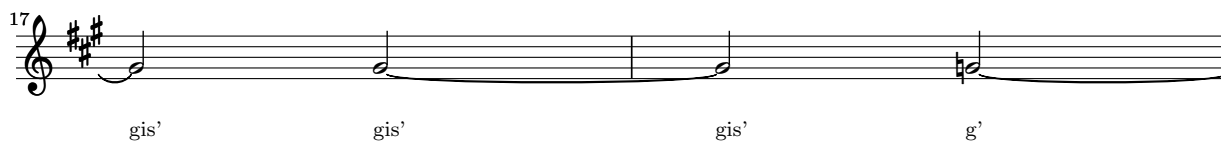


`'tie-accidental.ly':`

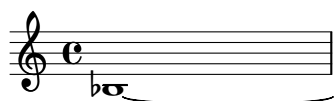
When tying notes with accidentals across a bar boundary, the accidental must not be drawn on the note in the next bar. Unless the tie crosses a line break, in which case the accidental is repeated if it would be different from an untied note. The next note of the same pitch in this next bar should always show the accidental (even if it's natural). Slurring a accidentaled note to a natural one across bar boundaries should be explicit.

Pitches can be verified by printing them with the `NoteNames` context.

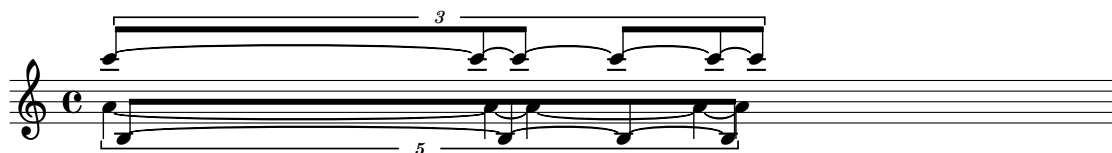




First and second bes (tied across line break) should get an accidental, but others should not. Only first B should get natural sign.

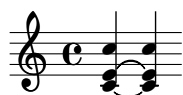


Tie engraver uses `busyGrobs` to keep track of note heads. Test if this queue works by throwing many mixed tuplets at it.



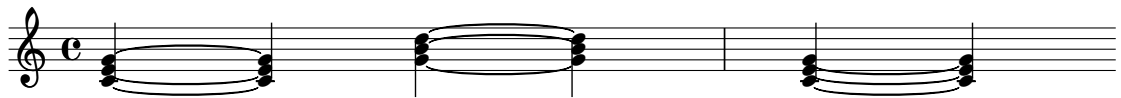
'tie-chord-partial.ly':

Tieing only parts of chords is possible. It requires putting the Tie engraver at Thread level, and redirecting untied notes to a different thread.



'tie-chord.ly':

When tying chords, the outer slurs point outwards, the inner slurs point away from the center of the staff. Override with `tieVerticalDirection`.



`'tie-dots.ly':`

Ties should not collide with dots.



`'tie-grace.ly':`

Tying a grace to the to a following grace or main note works.



`'tie.ly':`

Ties are strictly horizontal. They are placed in between note heads. The horizontal middle should not overlap with a staffline.



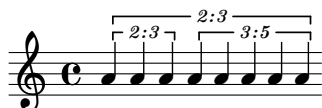
`'tuplet-beam.ly':`

In combination with a beam, the bracket of the tuplet bracket is removed. This only happens if there is one beam, as long as the bracket.



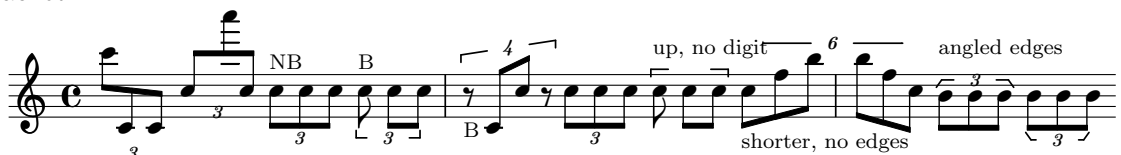
`'tuplet-nest.ly':`

Manual hack for nested tuplets, move outer tuplet up.



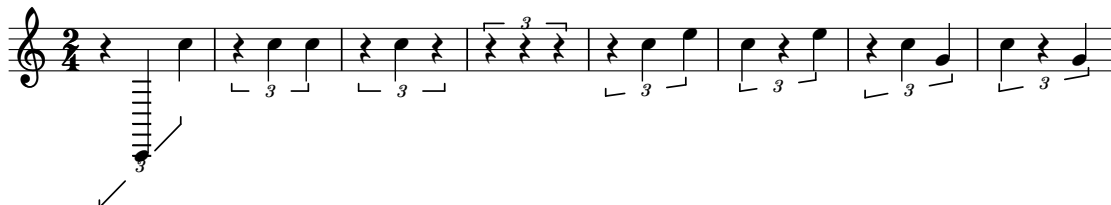
`'tuplet-properties.ly':`

Tuplet bracket formatting supports numerous options: NB should have no bracket, B should have bracket.



`'tuplet-rest.ly':`

Tests tuplet rests.



`'tuplet-staffline-collision.ly':`

Horizontal tuplet brackets are shifted vertically to avoid staff line collisions.



`'tuplets.ly':`

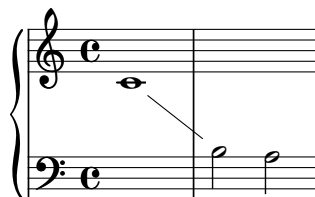
Tuplets are indicated by a bracket with a number. There should be no bracket if there is a beam exactly matching the length of the tuplet. The bracket does not interfere with the stafflines, and the number is centered in the gap in the bracket.

The bracket stops at the end of the stems, if the stems have the same direction as the bracket. The endings can be adjusted with `bracket-flare`.



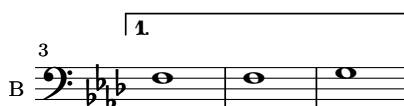
`'voice-follower.ly':`

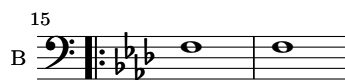
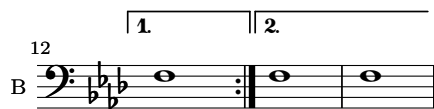
Whenever a voice switches to another staff a line connecting the notes can be printed automatically. This is enabled if the property `Thread.followVoice` is set to true.



`'volta-broken-left-edge.ly':`

Broken volta spanners behave correctly at left edge in all cases.





‘volta-multi-staff.ly’:

By setting `voltaOnThisStaff`, repeats can be put on more staves in a score.

