Nuove funzionalità nella versione 2.20 rispetto alla 2.18

Novità nella notazione musicale

Miglioramenti relativi alle altezze

- Le altezze il cui nome contiene un diesis o bemolle ora devono usare il trattino:

  ```
  \key a-flat \major
  ```

  invece di:

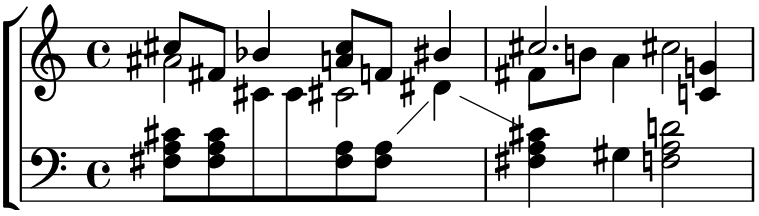
  ```
  \key aflat \major
  ```

  Non è invece necessario un secondo trattino per le altezze il cui nome contiene *doppi* diesis o bemolli. Per esempio, usando la notazione olandese `cisis`:

  ```
  \key c-sharpsharp \major
  ```

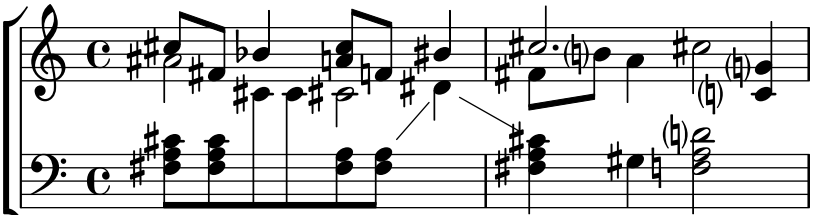
- Le regole delle alterazioni possono ora essere definite *attraverso* i contesti `ChoirStaff`.
- Sono state aggiunte due nuove regole delle alterazioni. Entrambe combinano le caratteristiche di `modern-voice`, `piano` e i loro equivalenti:

  `choral`

  Questo è ora lo stile di alterazioni predefinito per `ChoirStaff`.

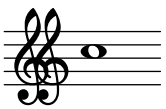
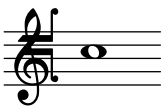
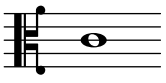
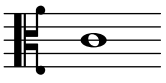
  `choral-cautionary`

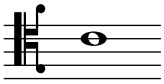
  Uguale a `choral` ma con l'aggiunta delle alterazioni di cortesia.

  Vedi anche Sezione "Alterazioni automatiche" in *Guida alla Notazione*.

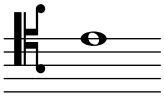
- Sono stati aggiunti quattro nuovi glifi di chiave: "GG" (doppio Sol), "Sol tenore", "varDo" e la relativa tessitura e "varpercussion":

Esempio	**Output**	**Esempio**	**Output**
`\clef GG`		`\clef tenorG`	
`\clef varC`		`\clef altovarC`	

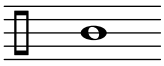
```
\clef tenorvarC
```



```
\clef baritonevarC
```



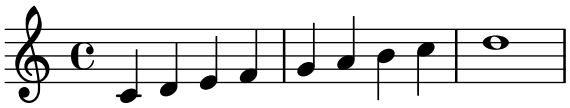
```
\clef varpercussion
```



Vedi anche Sezione “Stili della chiave” in *Guida alla Notazione*.

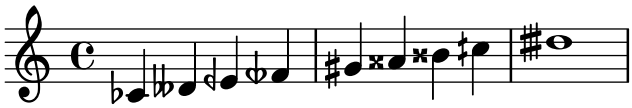
- I nomi francesi delle note ora sono definiti separatamente invece di essere riprese (come alias) dai nomi italiani. L’altezza *d* può essere inserita come `re` o `ré`:

```
\language "français"
do ré mi fa | sol la si do | ré1
```



I diesis doppi possono si inseriscono col suffisso `x`:

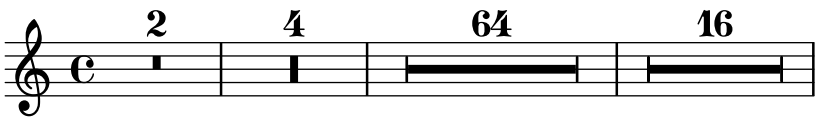
```
\language "français"
dob, rebb misb fabsb | sold ladd six dosd | rédsd1
```



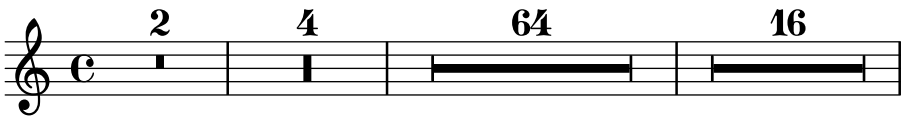
Miglioramenti relativi al ritmo

- Lo spazio orizzontale occupato dalle pause multiple è proporzionale alla loro durata totale e può essere modificato attraverso la proprietà `MultiMeasureRest.space-increment`. Il valore predefinito è `2.0`.

```
\compressFullBarRests
R1*2 R1*4 R1*64 R1*16
```



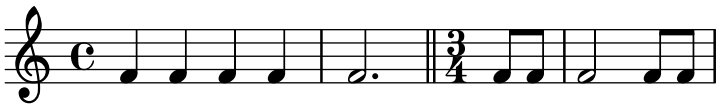

```
\compressFullBarRests
\override Staff.MultiMeasureRest.space-increment = 2.5
R1*2 R1*4 R1*64 R1*16
```



- Sono stati introdotti dei miglioramenti al comando `\partial` quando si usa con musica parallela e/o contesti multipli.
- È ora possibile cambiare l’indicazione di tempo a metà misura usando insieme `\time` e `\partial`:

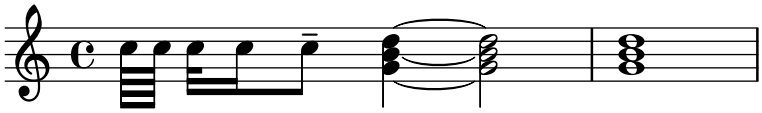
```
f f f f | f2. \bar "||"
```

```
\time 3/4 \partial 4
f8 8 | f2 f8 f |
```

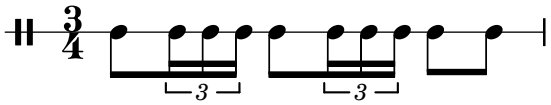


- Le durate isolate di una sequenza musicale ora sono interpretate come note prive di altezza. Le altezze vengono prese dalla nota o accordo precedenti. Ciò può essere utile specialmente per specificare le durate musicali sia nella musica che in funzioni Scheme e può aiutare a rendere più scorrevole la lettura dei file sorgente di LilyPond:

```
c64[ 64] 32 16 8^- <g b d>4~ 2 | 1
```



```
\new DrumStaff \with { \override StaffSymbol.line-count = 1 }
\drummode {
  \time 3/4
  tambourine 8 \tuplet 3/2 { 16 16 16 }
             8 \tuplet 3/2 { 16 16 16 } 8 8 |
}
```

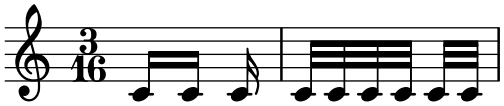


- Le eccezioni della disposizione delle travature possono essere costruite con la semplice funzione Scheme `\beamExceptions`. Prima bisognava scrivere:

```
\set Timing.beamExceptions =
#'(                          ;inizio della lista associativa
   (end .                    ;elemento per la chiusura delle travature
    (                        ;inizio della lista associativa per le estremità
     ((1 . 32) . (2 2 2))    ;regola per le travature di 1/32 -- chiudi ogni 1/16
    )))
\time #'(2 1) 3/16
c16 c c
\repeat unfold 6 { c32 }
```

Con la nuova funzione Scheme `beamExceptions`, questo frammento diventa:

```
\set Timing.beamExceptions =
  \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }

\time #'(2 1) 3/16
c16 c c |
\repeat unfold 6 { c32 } |
```

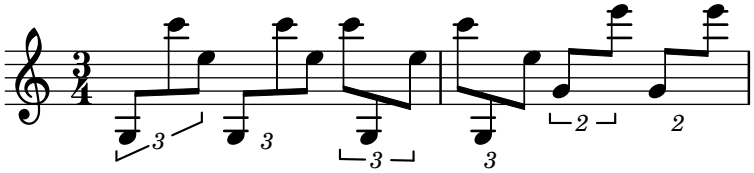


separando le molteplici eccezioni con i segni di controllo di battuta `|`. Scrivere lo schema ritmico senza altezza è comodo ma non è obbligatorio (vedi anche il miglioramento ritmico

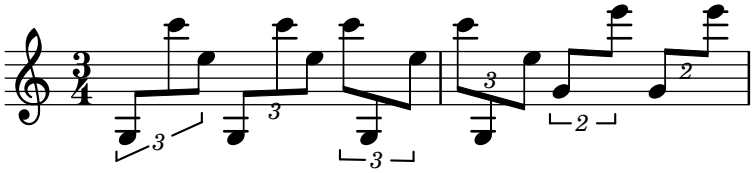
menzionato prima – *Le durate isolate di una sequenza musicale ora sono interpretate come note prive di altezza*).

- Il posizionamento dei numeri dei gruppi irregolari con travature angolari è stato migliorato molto. In precedenza, i numeri erano posizionati in base alla posizione della parentesi del gruppo irregolare, anche se questa era omessa. Ciò poteva causare numeri mal posizionati.

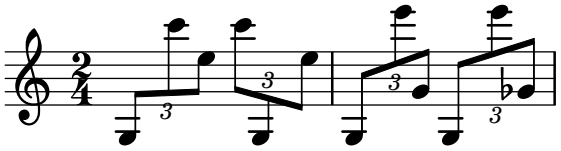
  Prima:



  Ora, quando la parentesi è omessa, i numeri sono posizionati più vicini:

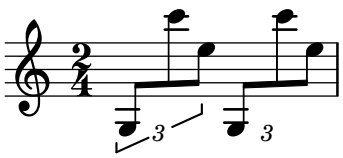


- È stato aggiunto anche il rilevamento delle collisioni per i numeri dei gruppi irregolari con travature angolari, per cui il numero viene spostato orizzontalmente se troppo vicino a una colonna di note adiacente (ma viene preservata la distanza verticale tra il numero e la travatura angolare). In caso di collisione, per esempio con un'alterazione, il numero del gruppo irregolare viene invece spostato verticalmente. Se il numero è troppo grande per entrare nello spazio disponibile, viene usato il sistema di posizionamento originale basato sulla parentesi.



  Il comportamento originale è ancora disponibile grazie a una nuova proprietà `knee-to-beam` dell'oggetto di formattazione `TupletNumber`:

```
\time 2/4
\override Beam.auto-knee-gap = 3
\override TupletNumber.knee-to-beam = ##f
\override TupletBracket.bracket-visibility = ##t
\tuplet 3/2 4 { g8 c'' e, }
\once \override TupletBracket.bracket-visibility = ##f
\tuplet 3/2 4 { g,,8 c'' e, }
```

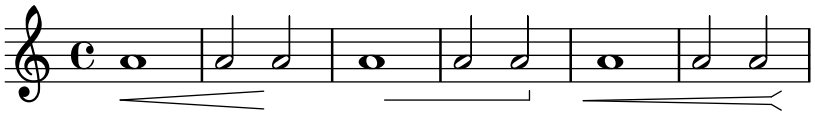


Miglioramenti dei segni espressivi

- Le estremità delle forcelle possono ora essere aggiustate precisamente tramite la proprietà `shorten-pair`, che precedentemente agiva soltanto sugli estensori del testo, come `TupletBracket` e `OttavaBracket`.

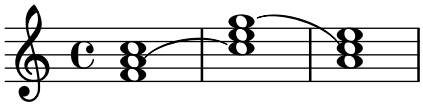
Valori positivi spostano a destra, quelli negativi a sinistra.

```
\once \override Hairpin.shorten-pair = #'(0 . 2)
a1\< | a2 a\!

\once \override Hairpin.shorten-pair = #'(2 . 0)
\once \override Hairpin.stencil = #constante-hairpin
a1\< | a2 a\!

\once \override Hairpin.shorten-pair = #'(-1 . -1)
\once \override Hairpin.stencil = #flared-hairpin
a1\< | a2 a\!
```

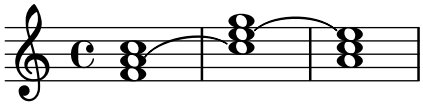


- Le legature di portamento e di frase possono ora iniziare da note individuali di un accordo.

```
<f a( c>1 | <c') e g(> | <a c) e>
```



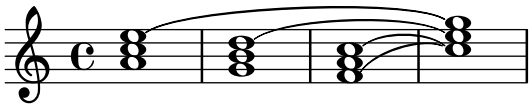
```
<f( a\( c>1 | <c'\) e\( g> | <a c e\)>
```



- È stato aggiunto un nuovo comando, `\=X`, dove 'X' può essere qualsiasi simbolo o numero intero non negativo, in modo che un 'id' specifico possa essere assegnato all'inizio e alla fine delle legature di portamento e di frase.

  È utile quando sono richieste legature di portamento simultanee o se una legatura si sovrappone a un'altra oppure quando si annidano brevi legature all'interno di una più lunga.

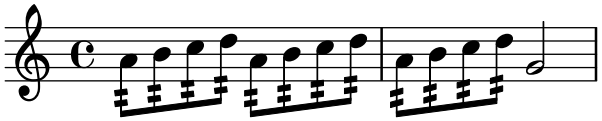
```
<a c e\=7\(>1 | <g b d\=£(> |
<f\=A( a c\="foo"(> | <c'\="foo")\=A) e\=£) g\=7\)> |
```



Vedi anche Sezione "Indicazioni espressive curvilinee" in *Guida alla Notazione*.

Miglioramenti della notazione delle ripetizioni

- Lo stile visivo delle barre del tremolo (forma, stile e inclinazione) è ora regolato in modo più preciso.



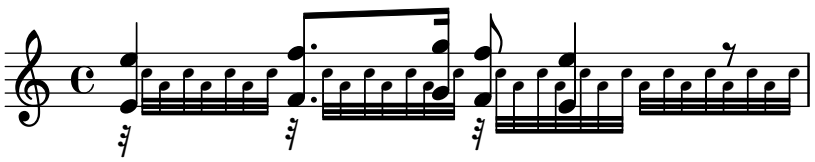
- La funzione musicale `\unfoldRepeats` può ora prendere una lista di argomenti (facoltativa) che specifica quali tipi di musica ripetuta debbano essere ricopiati. Le opzioni possibili

sono `percent`, `tremolo` e `volta`. Se la lista facoltativa non viene specificata, verrà usato `repeated-music`, che ricopia tutto.

Miglioramenti della notazione del rigo

- È stato aggiunto il nuovo comando `\magnifyStaff` che scala la dimensione del rigo, delle linee del rigo, delle stanghette, delle lineette della travatura e della spaziatura orizzontale generalmente al livello di contesto `Staff`. Le linee del rigo non vengono ridotte a una dimensione inferiore a quella predefinita perché lo spessore di gambi, legature e simili è basato sullo spessore della linea del rigo.
- È stato aggiunto un nuovo comando `\magnifyMusic`, che permette di cambiare la dimensione della notazione senza cambiare la dimensione del rigo, ridimensionando proporzionalmente in automatico i gambi, le travature e la spaziatura orizzontale.

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c'' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
>>
```



- È disponibile un nuovo comando, `\RemoveAllEmptyStaves`, che si comporta proprio come `\RemoveEmptyStaves`, con la differenza che toglie anche i righi vuoti del primo sistema di una partitura.
- È stato aggiunto un nuovo comando per il testo: `\justify-line`. È simile al comando `\fill-line` con la differenza che invece di impostare le *parole* in colonne, il comando `\justify-line` bilancia lo spazio tra di esse assicurando che sia sempre regolare se ci sono tre o più parole nel testo.

```
\markup \fill-line {oooooo oooooo oooooo oooooo}
\markup \fill-line {ooooooooo ooooooooo oo ooo}
```

oooooo oooooo oooooo oooooo

ooooooooo oooooooo oo ooo

```
\markup \justify-line {oooooo oooooo oooooo oooooo}
\markup \justify-line {ooooooooo ooooooooo oo ooo}
```

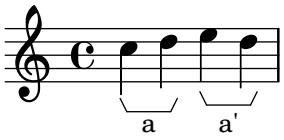
oooooo oooooo oooooo oooooo

ooooooooo ooooooooo oo ooo

Miglioramenti delle annotazioni editoriali

- È ora possibile aggiungere del testo alle parentesi per l'analisi musicale tramite l'oggetto `HorizontalBracketText`.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}

{
  \once \override HorizontalBracketText.text = "a"
  c''\startGroup d''\stopGroup
  e''-\tweak HorizontalBracketText.text "a'" \startGroup d''\stopGroup
}
```



Miglioramenti alla formattazione del testo

- Ora è più facile usare font 'musicali' alternativi al predefinito Emmentaler in LilyPond. Maggiori informazioni in Sezione "Cambiare il tipo di carattere della notazione" in *Guida alla Notazione*.
- Il font predefinito del testo è cambiato da `Century Schoolbook L`, `sans-serif`, e `monospace`.

  Per il backend `svg`:

Famiglia	**Font predefinito**
roman	`serif`
sans	`sans-serif`
typewriter	`monospace`

  `serif`, `sans-serif`, e `monospace` sono `generic-family` (famiglie generiche) nelle specifiche SVG e CSS.

  Per gli altri backend:

Famiglia	**Font predefinito (alias)**	**Elenchi di definizione di alias**
roman	`LilyPond Serif`	TeX Gyre Schola, C059, Century SchoolBook URW, Century Schoolbook L, DejaVu Serif, ..., serif
sans	`LilyPond Sans Serif`	TeX Gyre Heros, Nimbus Sans, Nimbus Sans L, DejaVu Sans, ..., sans-serif
typewriter	`LilyPond Monospace`	TeX Gyre Cursor, Nimbus Mono PS, Nimbus Mono, Nimbus Mono L, DejaVu Sans Mono, ..., monospace

  `LilyPond Serif`, `LilyPond Sans Serif`, e `LilyPond Monospace` sono font alias definiti nel file di configurazione FontConfig specifico per LilyPond `00-lilypond-fonts.conf`. Se un carattere non esiste nel primo font elencato, il font successivo dell'elenco verrà usato al suo posto per quel carattere. I dettagli delle definizioni degli alias si trovano nel file `00-lilypond-fonts.conf` all'interno della directory di installazione.

- Quando si usano i font OpenType, sono disponibili le funzionalità dei font. Nota: non tutti i font OpenType hanno tutte le funzioni.

```
% Maiuscoletto vero
\markup { Stile normale: Hello HELLO }
\markup { \caps { Maiuscoletto: Hello } }
\markup { \override #'(font-features . ("smcp"))
          { Maiuscoletto vero: Hello } }

% Stili numerici
\markup { Stile numerico normale: 0123456789 }
\markup { \override #'(font-features . ("onum"))
          { Stile numerico vecchio: 0123456789 } }

% Alternative stilistiche
\markup { \override #'(font-features . ("salt 0"))
          { Alternative stilistiche 0: ϵϕπρθ } }
\markup { \override #'(font-features . ("salt 1"))
          { Alternative stilistiche 1: ϵϕπρθ } }

% Funzionalità multiple
\markup { \override #'(font-features . ("onum" "smcp" "salt 1"))
          { Funzionalità multiple: Hello 0123456789 ϵϕπρθ } }
```

Stile normale: Hello HELLO

Maiuscoletto: Hello

Maiuscoletto vero: Hello

Stile numerico normale: 0123456789

Stile numerico vecchio: 0123456789

Alternative stilistiche 0: ϵφπρθ

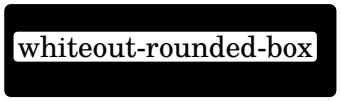
Alternative stilistiche 1: ϵϕϖϱϑ

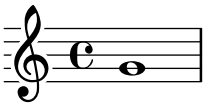
Funzionalità multiple: Hello 0123456789 ϵϕϖϱϑ

- Sono ora disponibili due nuovi stili di whiteout (bianchetto). Lo stile `outline` approssima i contorni del profilo di un glifo e la sua forma è prodotta da varie copie sovrapposte del glifo. Lo stile `rounded-box` genera una forma rettangolare stondata. Per tutti e tre gli stili, incluso lo stile predefinito `box`, lo spessore (`thickness`) della forma di whiteout può essere personalizzato come multiplo dello spessore della linea del rigo.

```
\markup {
  \combine
    \filled-box #'(-1 . 15) #'(-3 . 4) #1
    \override #'(thickness . 3)
    \whiteout whiteout-box
}
```

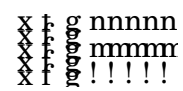
```
\markup {
  \combine
    \filled-box #'(-1 . 24) #'(-3 . 4) #1
    \override #'(style . rounded-box)
    \override #'(thickness . 3)
    \whiteout whiteout-rounded-box
}
\markup {
  \combine
    \filled-box #'(-1 . 18) #'(-3 . 4) #1
    \override #'(style . outline)
    \override #'(thickness . 3)
    \whiteout whiteout-outline
}
\relative {
  \override Staff.Clef.whiteout-style = #'outline
  \override Staff.Clef.whiteout = 3
  g'1
}
```





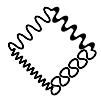
- Un nuovo comando di tipo markup, `\with-dimensions-from`, semplifica l'uso di `\with-dimensions` prendendo le nuove dimensioni da un oggetto di markup, indicato come primo argomento.

```
\markup {
  \pattern #5 #Y #0 "x"
  \pattern #5 #Y #0 \with-dimensions-from "x" "f"
  \pattern #5 #Y #0 \with-dimensions-from "x" "g"
  \override #'(baseline-skip . 2)
  \column {
    \pattern #5 #X #0 "n"
    \pattern #5 #X #0 \with-dimensions-from "n" "m"
    \pattern #5 #X #0 \with-dimensions-from "n" "!"
  }
}
```



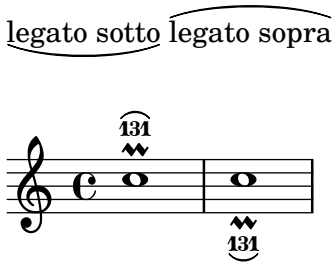
- È ora disponibile il comando markup `\draw-squiggle-line`. È possibile personalizzarlo modificando le proprietà `thickness`, `angularity`, `height` e `orientation`

```
\markup
  \overlay {
    \draw-squiggle-line #0.5 #'(3 . 3) ##t

    \translate #'(3 . 3)
    \override #'(thickness . 4)
    \draw-squiggle-line #0.5 #'(3 . -3) ##t

    \translate #'(6 . 0)
    \override #'(angularity . -5)
    \draw-squiggle-line #0.5 #'(-3 . -3) ##t

    \translate #'(3 . -3)
    \override #'(angularity . 2)
    \override #'(height . 0.3)
    \override #'(orientation . -1)
    \draw-squiggle-line #0.2 #'(-3 . 3) ##t
  }
```



- Oltre al comando markup generico `\tie`, sono ora disponibili i comandi markup `\undertie` e `\overtie`.

```
\markup {
  \undertie "legato sotto"
  \overtie "legato sopra"
}

m = {
  c''1 \prall -\tweak text \markup \tie "131" -1
}

{ \voiceOne \m \voiceTwo \m }
```



Novità per la notazione specialistica

Miglioramenti della musica vocale

- Viene fornito un nuovo e flessibile modello per vari tipi di musica corale. Può essere usato per creare semplice musica corale, con o senza accompagnamento per pianoforte, in due o quattro righi. Diversamente da altri modelli, questo modello è 'integrato', ovvero non c'è

bisogno di copiarlo e modificarlo: basta includerlo con `\include` nel file di input. Maggiori dettagli in Sezione "Modelli integrati" in *Manuale di Apprendimento*.

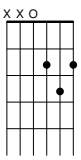
- La funzione `\addlyrics` ora funziona con contesti arbitrari incluso `Staff`.
- `\lyricsto` e `\addLyrics` sono stati 'armonizzati'. Entrambi ora accettano lo stesso tipo di lista di argomenti limitata che accettano anche `\lyrics` e `\chords`. È stata aggiunta la compatibilità all'indietro così che gli identificatori della musica (es: `\mus`) sono permessi come argomenti. È stata aggiunta a convert-ly una regola che toglie gli usi ridondanti di `\lyricmode` e riorganizza le combinazioni con l'inizio dei contesti in modo che `\lyricsto` in generale sia applicato per ultimo (ovvero come accadrebbe con `\lyricmode`).

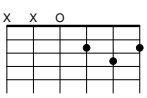
Miglioramenti relativi agli strumenti a corda con e senza tasti

- È stato aggiunto un nuovo stile per le teste di nota dell'intavolatura: `TabNoteHead.style = #'slash`.
- Nei diagrammi dei tasti la distanza tra i tasti e quella tra le corde sono ora regolabili in modo indipendente. Sono disponibili `fret-distance` e `string-distance` come sottoproprietà di `fret-diagram-details`.

```
fretMrkp = \markup { \fret-diagram-terse "x;x;o;2;3;2;" }

\markuplist
\override #'(padding . 2)
\table #'(0 -1) {
  "predefinito"

  \fretMrkp

  "fret-distance"

  \override #'(fret-diagram-details . ((fret-distance . 2)))
  \fretMrkp

  "string-distance"

  \override #'(fret-diagram-details . ((string-distance . 2)))
  \fretMrkp
}
```

predefinito 

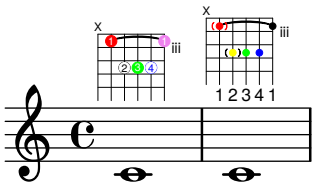
fret-distance 

string-distance 

- È ora possibile colorare individualmente sia i punti che le parentesi nei diagrammi dei tasti quando si usa il comando `\fret-diagram-verbose` dentro un blocco `\markup`.

```
\new Voice {
```

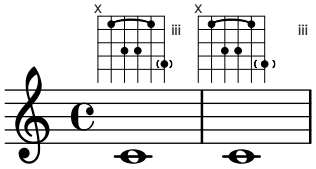
```
    c1^\markup {
      \override #'(fret-diagram-details . (
                   (finger-code . in-dot))) {
        \fret-diagram-verbose #'((mute 6)
                                 (place-fret 5 3 1 red)
                                 (place-fret 4 5 2 inverted)
                                 (place-fret 3 5 3 green)
                                 (place-fret 2 5 4 blue inverted)
                                 (place-fret 1 3 1 violet)
                                 (barre 5 1 3 ))
      }
    }
    c1^\markup {
      \override #'(fret-diagram-details . (
                   (finger-code . below-string))) {
        \fret-diagram-verbose #'((mute 6)
                                 (place-fret 5 3 1 red parenthesized)
                                 (place-fret 4 5 2 yellow
                                             default-paren-color
                                             parenthesized)
                                 (place-fret 3 5 3 green)
                                 (place-fret 2 5 4 blue )
                                 (place-fret 1 3 1)
                                 (barre 5 1 3))
      }
    }
}
```



- Sono state aggiunte due nuove proprietà da usare in `fret-diagram-details` quando sia usa il comando `\fret-diagram-verbose` in un blocco markup; `fret-label-horizontal-offset`, che agisce su `fret-label-indication`, e `paren-padding` che regola lo spazio tra il punto e le parentesi che lo circondano.

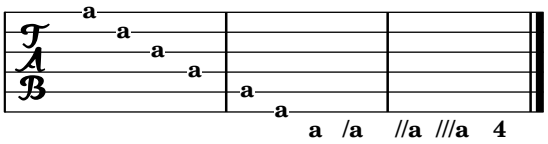
```
\new Voice {
  c1^\markup {
    \fret-diagram-verbose #'((mute 6)
                             (place-fret 5 3 1)
                             (place-fret 4 5 2)
                             (place-fret 3 5 3)
                             (place-fret 1 6 4 parenthesized)
                             (place-fret 2 3 1)
                             (barre 5 2 3))
  }
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (fret-label-horizontal-offset . 2)
                 (paren-padding . 0.25))) {
```

```
      \fret-diagram-verbose #'((mute 6)
                               (place-fret 5 3 1)
                               (place-fret 4 5 2)
                               (place-fret 3 5 3)
                               (place-fret 1 6 4 parenthesized)
                               (place-fret 2 3 1)
                               (barre 5 2 3))
    }
  }
}
```



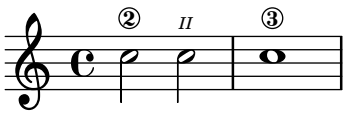
- Sono supportate ulteriori corde di basso (per l'intavolatura per liuto).

```
m = { f'4 d' a f d a, g, fis, e, d, c,  \bar "|." }

\score {
  \new TabStaff \m
  \layout {
    \context {
      \Score
      tablatureFormat = #fret-letter-tablature-format
    }
    \context {
      \TabStaff
      stringTunings = \stringTuning <a, d f a d' f'>
      additionalBassStrings = \stringTuning <c, d, e, fis, g,>
      fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k")
    }
  }
}
```



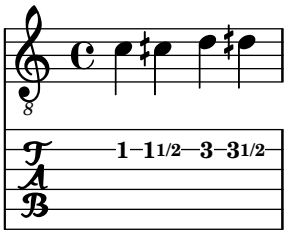
- I numeri di corda ora possono essere stampati in numeri romani (per esempio, per gli strumenti a corda senza tasti).

```
c2\2
\romanStringNumbers
c\2
\arabicStringNumbers
c1\3
```



- `TabStaff` è ora capace di mostrare i microtoni, utili per il bending etc.

```
\layout {
  \context {
    \Score
    supportNonIntegerFret = ##t
  }
}

mus = \relative { c'4 cih d dih }

<<
  \new Staff << \clef "G_8" \mus >>
  \new TabStaff \mus
>>
```



Miglioramenti alla notazione per accordo

- `\chordmode` può ora usare i costrutti `< >` e `<< >>`.
- È ora possibile sovrascrivere la proprietà `text` dei nomi degli accordi.

```
<<
\new ChordNames \chordmode {
  a' b c:7
  \once \override ChordName.text = "bla"
  d
}
>>
```

Novità per input e output

Miglioramenti della struttura di input

- I blocchi introdotti con `\header` possono essere salvati in variabili e usati come argomenti di funzioni musicali e funzioni Scheme e come parte dei costrutti `#{...#}`. Sono rappresentati come un modulo Guile.

  Sebbene i blocchi `\book`, `\bookpart`, `\score`, `\with`, `\layout`, `\midi`, `\paper` possano essere trasferiti in un modo simile, sono tuttavia rappresentati da tipi di dati diversi.

Miglioramenti a titoli e intestazioni

- I numeri di pagina ora possono essere stampati in numeri romani impostando la variabile del foglio `page-number-type`.

Miglioramenti al file di input

- È stato aggiunto un nuovo comando `\tagGroup`, che si aggiunge a quelli esistenti `\keepWithTag` e `\removeWithTag`. Per esempio:

```
\tagGroup #'(violinI violinII viola cello)
```

dichiara una lista di 'etichette' (*tag*) che appartiene a un solo 'gruppo di etichette'.

```
\keepwithTag #'violinI
```

ora si preoccupa solo delle 'etichette' del gruppo cui appartiene l'etichetta 'violinI'.

Qualsiasi elemento della musica inclusa contrassegnato con una o più etichette del gruppo, ma *non* con *violinI*, sarà rimosso.

Miglioramenti dell'output

- I file sorgente LilyPond ora possono essere incorporati nei file PDF generati. Questa funzionalità sperimentale è disabilitata per impostazione predefinita e può essere considerata non sicura, dato che documenti PDF con del contenuto nascosto tendono a costituire un rischio di sicurezza. Non tutti i lettori PDF sono capaci di gestire i documenti incorporati (in questo caso, l'output PDF apparirà normalmente e i file sorgente resteranno invisibili). Questa funzionalità funziona solo col backend PDF.
- Sono ora disponibili la procedura `output-classic-framework` e l'opzione `-dclip-systems` per il backend `SVG`.
- È stato aggiunto un argomento, `-dcrop`,che formatta l'output `SVG` e `PDF` senza margini né interruzioni di pagina.
- Per l'output SVG viene ora utilizzata la nuova proprietà grob `output-attributes` al posto della proprietà `id`. Permette di definire molteplici attributi come una lista associativa. Per esempio, `#'((id . 123) (class . foo) (data-blabla . \bar"))` produrrà il seguente elemento gruppo (g) in un file SVG: `<g id=\123" class=\foo" data-blabla=\bar"> ... </g>`.
- La funzionalità PostScript di regolazione del tratto non è più applicata automaticamente bensì è lasciata alla discrezione del dispositivo PostScript (il comportamento predefinito di Ghostscript è di usarla per risoluzioni fino a 150ppp quando genera immagini raster). Se abilitata, viene utilizzato un algoritmo di disegno più complesso per avvantaggiarsi della regolazione del tratto.

  La regolazione del tratto può essere forzata specificando l'opzione da linea di comando '`-dstrokeadjust`' dell'eseguibile lilypond. Quando si generano file `PDF`, di solito ciò produce anteprime `PDF` notevolmente migliori ma anche file di dimensioni maggiori. La qualità della stampa ad alte risoluzioni non è interessata da questa modifica.
- Aggiunta una nuova funzione `make-path-stencil` che supporta tutti i comandi `path` sia relativi che assoluti:

  `lineto`, `rlineto`, `curveto`, `rcurveto`, `moveto`, `rmoveto`, `closepath`. La funzione supporta anche la sintassi di 'single-letter' usata nei comandi path standard dei file SVG:

  `L`, `l`, `C`, `c`, `M`, `m`, `Z` e `z`. Il nuovo comando è anche compatibile all'indietro con la funzione originale `make-connected-path-stencil`. Si veda anche `scm/stencil.scm`.

Miglioramenti relativi al MIDI

- Le articolazioni più comuni sono ora presenti nell'output MIDI. L'accento e il marcato aumentano il volume delle note; staccato, staccatissimo e portato le rendono più brevi. I respiri abbreviano la nota precedente.

  Tale comportamento può essere personalizzato attraverso le proprietà `midiLength` e `midiExtraVelocity` in `ArticulationEvent`. Si vedano gli esempi in `script-init.ly`.
- Migliorato l'output MIDI dei respiri. Dopo le note legate con legatura di valore, i respiri prendono il tempo *solo* dall'ultima nota della legatura; per esempio, `{ c4~ c8 \breathe }` viene riprodotto come `{ c4~ c16 r }` invece di `{ c4 r8 }`. Ciò è più coerente con le articolazioni e col modo in cui l'essere umano interpreta i respiri che seguono una legatura di valore. Semplifica anche l'allineamento di respiri simultanei su molteplici parti, tutte con diverse lunghezze delle note.

- È ora possibile regolare il 'livello di espressione' dei canali MIDI usando la proprietà di contesto `Staff.midiExpression`. Si può usare per alterare il volume percepito delle note sostenute in modo uniforme (sebbene in un modo molto di 'basso livello'); si può specificare un valore compreso tra `0.0` e `1.0`.

```
\score {
  \new Staff \with {
    midiExpression = #0.6
    midiInstrument = "clarinet"
  }
  <<
    { a'1~ a'1 }
    {
      \set Staff.midiExpression = #0.7 s4\f\<
      \set Staff.midiExpression = #0.8 s4
      \set Staff.midiExpression = #0.9 s4
      \set Staff.midiExpression = #1.0 s4

      \set Staff.midiExpression = #0.9 s4\>
      \set Staff.midiExpression = #0.8 s4
      \set Staff.midiExpression = #0.7 s4
      \set Staff.midiExpression = #0.6 s4\!
    }
  >>
  \midi { }
}
```

- Quando crea il file MIDI, LilyPond ora salva il titolo (`title`) definito nel blocco `\header` di una partitura (o, se tale definizione non è presente a livello di `\score`, la prima definizione trovata in un blocco `\header` del blocco `\bookpart`, `\book` o del livello superiore) come nome della sequenza MIDI nel file MIDI. Il nome della sequenza MIDI può anche essere definito tramite il nuovo campo `miditítle` del blocco `\header`, che ha priorità sul campo `title` (ciò può essere utile, per esempio, se `title` contiene della formattazione che non può essere resa automaticamente in testo semplice in modo soddisfacente).
- È stato aggiunto il supporto per semplificare l'uso di font 'musicali' alternativi al font predefinito Emmentaler. Maggiori informazioni in Sezione "Cambiare il tipo di carattere della notazione" in *Guida alla Notazione*.

Miglioramenti dell'estrazione di musica

- `\displayLilyMusic` e le sue sottostanti funzioni Scheme non omettono più le durate ridondanti. Ciò semplifica il riconoscimento affidabile e la formattazione delle durate isolate in espressioni come questa

```
{ c4 d4 8 }
```

Novità relative alla spaziatura

Miglioramenti dell'interruzione di pagina

- Ci sono due nuove funzioni di interruzione della pagina. `ly:one-page-breaking` modifica automaticamente l'altezza della pagina per far entrare la musica, in modo che stia tutta in una pagina. `ly:one-line-auto-height-breaking` è simile a `ly:one-line-breaking`, perché posiziona la musica su una sola linea regolando la larghezza della pagina, tuttavia modifica automaticamente anche l'altezza della pagina per farci entrare la musica.

Miglioramenti della spaziatura verticale e orizzontale

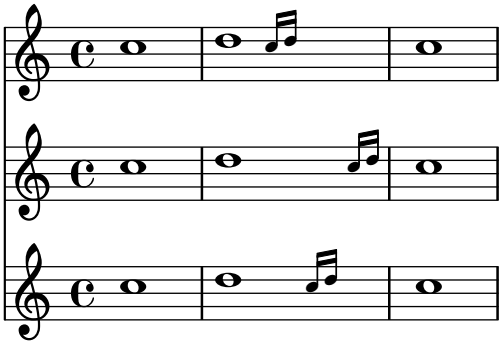
- È ora possibile spostare i sistemi rispetto alla loro posizione corrente tramite `extra-offset`, sottoproprietà di `NonMusicalPaperColumn.line-break-system-details`. Sono consentite modifiche sia verticali che orizzontali. Questa funzionalità è utile in particolare per fare piccoli aggiustamenti della posizione verticale predefinita dei singoli sistemi. Maggiori informazioni in Sezione "Posizionamento esplicito di righi e sistemi" in *Guida alla Notazione*.
- Migliorata la spaziatura verticale delle teste, di dimensione piccola e normale, della nota 'MI' negli stili Funk e Walker, così che ora abbiano la stessa larghezza di altre note a forma variabile nei loro rispettivi gruppi. Anche le teste della nota `SOL` ora sono migliorate visivamente se utilizzate con le teste di dimensione normale o sottile degli stili Aiken e Sacred Harp.
- `LeftEdge` ora ha una proprietà `Y-extent` (verticale) che può essere definita. Si veda Sezione "LeftEdge" in *Guida al Funzionamento Interno*.
- I grob e i loro oggetti genitori possono essere allineati in modo separato consentendo più flessibilità nelle posizioni dei grob. Per esempio il margine 'sinistro' di un grob ora può essere allineato al 'centro' del suo oggetto genitore.
- Migliorato l'allineamento orizzontale quando si usa `TextScript`, con `DynamicText` o `LyricText`.

Novità a proposito di cambiamenti delle impostazioni predefinite

È stato aggiunto un argomento opzionale per il comando `\afterGrace`.

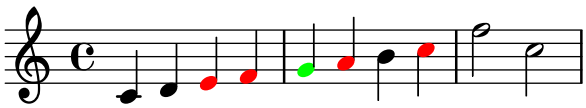
Ora `\afterGrace` ha un argomento opzionale per specificare tramite una frazione la posizione di spaziatura delle sue note.

```
<<
\new Staff \relative {
  % Il valore predefinito codificato nei sorgenti (3/4)
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  % Modifica manuale del valore predefinito codificato nei sorgenti (15/16)
  #(define afterGraceFraction (cons 15 16))
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  % Uso del nuovo argomento (5/6)
  c''1 \afterGrace 5/6 d1 { c16[ d] } c1
}
>>
```



- Tutti i comandi `\override`, `\revert`, `\set` e `\unset` ora funzionano col prefisso `\once`, rendendo possibili le impostazioni temporanee.

```
\relative {
  c'4 d
  \override NoteHead.color = #red
  e4 f |
  \once \override NoteHead.color = #green
  g4 a
  \once \revert NoteHead.color
  b c |
  \revert NoteHead.color
  f2 c |
}
```

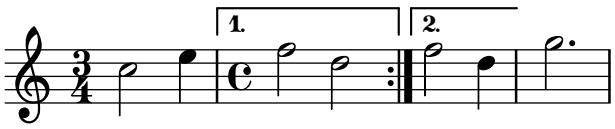


Novità a proposito di interfacce interne e funzioni

- La proprietà musicale e dei grob `spanner-id`, usata per distinguere legature di portamento simultanee e legature di frase, è stata modificata: non è più una stringa, bensì una 'key', ovvero un numero intero non negativo o un simbolo (vedi anche il miglioramento del segno espressivo citato prima – *il nuovo comando* \=*X*).
- Le proprietà di contesto nominate nella proprietà '`alternativeRestores`' sono ripristinate al loro valore presente all'inizio della *prima* alternativa in tutte le alternative successive.

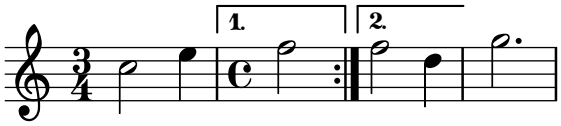
  Attualmente l'impostazione predefinita ripristina il 'metro corrente';

```
\time 3/4
\repeat volta 2 { c2 e4 | }
\alternative {
  { \time 4/4 f2 d | }
  { f2 d4 | }
}
g2. |
```



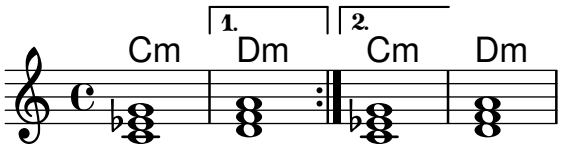
'la posizione della misura';

```
\time 3/4
\repeat volta 2 { c2 e4 | }
\alternative {
  { \time 4/4
    \set Timing.measurePosition = #(ly:make-moment -1/2)
    f2 | }
  { f2 d4 | }
}
g2. |
```



e i ‘cambi di accordo’:

```
<<
  \new ChordNames {
    \set chordChanges = ##t
    \chordmode { c1:m d:m c:m d:m }
  }
  \new Staff {
    \repeat volta 2 { \chordmode { c1:m } }
    \alternative {
      { \chordmode { d:m } }
      { \chordmode { c:m } }
    }
  \chordmode { d:m }
}
>>
```



- Le funzioni LilyPond definite con `define-music-function`, `define-event-function`, `define-scheme-function` e `define-void-function` ora possono essere richiamate direttamente da Scheme come se fossero vere procedure Scheme. Il controllo e la corrispondenza degli argomenti sono eseguiti sempre nello stesso modo come quando la funzione viene richiamata attraverso l’input di LilyPond. Ciò comprende l’inserimento dei valori predefiniti per gli argomenti opzionali che non corrispondono ai loro predicati. Invece di usare `\default` nella vera lista degli argomenti per saltare esplicitamente una sequenza di argomenti opzionali, si può usare `*unspecified*`.
- La posizione dell’input attuale e il decodificatore sono ora salvati nei “fluid” di Guile e possono essere citati attraverso le chiamate di funzione `(*location*)` e `(*parser*)`. Di conseguenza molte funzioni che prima richiedevano un argomento `parser` esplicito non ne hanno più bisogno.

  Le funzioni definite con `define-music-function`, `define-event-function`, `define-scheme-function` e `define-void-function` non usano più gli argomenti`parser` e `location`.

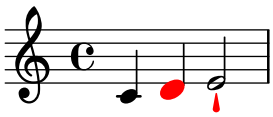
  Nel caso di queste definizioni in particolare, LilyPond cercherà di riconoscere l’uso obsoleto degli argomenti `parser` e `location`, fornendo per un po’ della semantica retrocompatibile.
- Le funzioni e gli identificatori Scheme ora possono essere usati come definizioni di output.
- Le espressioni Scheme possono ora essere usate come costituenti di un accordo.
- Le funzioni musicali (e quelle Scheme e vuote) e i comandi markup che forniscono soltanto i parametri finali a una catena di override e chiamate di funzioni musicali e comandi markup, ora possono essere definite semplicemente scrivendo l’espressione seguita da `\etc`.

```
\markup bold-red = \markup \bold \with-color #red \etc
highlight = \tweak font-size 3 \tweak color #red \etc

\markup \bold-red "text"
\markuplist \column-lines \bold-red { One Two }

{ c' \highlight d' e'2-\highlight -! }
```

text

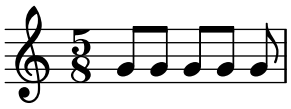
One

Two



- Le liste di simboli separate da punti come `FretBoard.stencil` sono supportate già dalla versione 2.18. Ora possono contenere anche numeri interi non negativi e possono essere separate anche con le virgole. Ciò permette di usare, per esempio:

```
{ \time 2,2,1 5/8  g'8 8 8 8 8 }
```



e

```
\tagGroup violin,oboe,bassoon
```

- Agli elementi delle liste associative potevano già essere assegnati dei valori individualmente (per esempio, variabili `\paper` come `system-system-spacing.basic-distance`). Ora possono anche essere citati nello stesso modo, come in questo esempio:

```
\paper {
  \void \displayScheme \system-system-spacing.basic-distance
}
```

  In combinazione con i cambiamenti precedentemente menzionati, ciò permette di impostare e citare pseudovariabili come `violin.1`.

- È ora disponibile il comando di tipo markup-list `\table`. Ogni colonna può essere allineata in modo diverso.

```
\markuplist {
    \override #'(padding . 2)
    \table
      #'(0 1 0 -1)
      {
        \underline { center-aligned right-aligned center-aligned left-aligned }
        one "1" thousandth "0.001"
        eleven "11" hundredth "0.01"
        twenty "20" tenth "0.1"
        thousand "1000" one "1.0"
      }
}
```

center-aligned	right-aligned	center-aligned	left-aligned
one	1	thousandth	0.001
eleven	11	hundredth	0.01

twenty	20	tenth	0.1
thousand	1000	one	1.0

- `InstrumentName` ora supporta l'interfaccia `text-interface`.
- La proprietà `thin-kern` del grob `BarLine` è stata rinominata `segno-kern`.
- I grob `KeyCancellation` ora ignorano le chiavi delle notine (come fanno anche i grob `KeySignature`).
- I grob `KeyCancellation` ora ignorano le chiavi in corpo piccolo (come fanno i grob `KeySignature`).
- Aggiunto il supporto per `\once \unset`