

## Nuove funzionalità nella versione 2.20 rispetto alla 2.18

- Le regole delle alterazioni possono essere definite nei contesti `ChoirStaff`. Sono disponibili due nuove regole, `choral` e `choral-cautionary`, che combinano le caratteristiche di `modern-voice` e `piano` o i loro equivalenti per le alterazioni di cortesia.
- 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.
- 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-whatever . \bar"))` produrrà il seguente elemento gruppo (g) in un file SVG: `<g id=\123" class=\foo" data-whatever=\bar"> ... </g>`.
- Le legature di portamento e di frase possono ora iniziare da note individuali di un accordo. Varie legature di portamento simultanee per ogni voce devono essere distinte con l'impostazione `spanner-id`.
- 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.
- È stato aggiunto il nuovo comando `\=` per specificare l'identificativo degli estensori, `spanner-id`, per le legature di portamento e di frase simultanee.

```
\fixed c' {  
  <c~ f\=1( g\=2( >2 <c e\=1) a\=2) >  
}
```



- 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.

- 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
```

- Queste liste possono essere usate anche nelle espressioni di assegnamenti, impostazioni (`\set`) e riscritture (`override`). Ciò permette di usare, per esempio:

```
{ \unset Timing.beamExceptions
```

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

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

|               |     |     |     |     |     |       |       |     |  |
|---------------|-----|-----|-----|-----|-----|-------|-------|-----|--|
| $\mathcal{T}$ | $a$ |     |     |     |     |       |       |     |  |
| $A$           | $a$ |     |     |     |     |       |       |     |  |
| $B$           | $a$ | $a$ |     |     |     |       |       |     |  |
|               |     | $a$ | $a$ |     |     |       |       |     |  |
|               |     |     | $a$ | $a$ | $a$ | $//a$ | $//a$ | $4$ |  |

- \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   |

- 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" "!"
  }
}

```

```

x f g nnnnn
x f g mmmmm
x f g !!!!!

```

- 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.
- 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
}

```



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

```

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

```

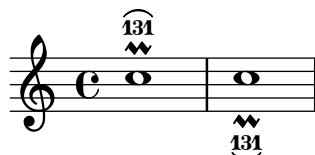
```

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

```

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

legato sotto legato sopra



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

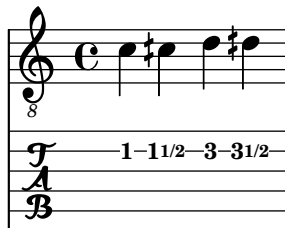
```

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

```

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

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



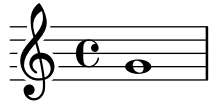
- Sono ora disponibili due nuovi stili di whiteout (bianchetto). Lo stile **outline** approssima i contorni del profilo di un glifo e la sua forma è prodotta da varie copie sovrapposte del glifo. Lo stile **rounded-box** genera una forma rettangolare stondata. Per tutti e tre gli stili, incluso lo stile 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
}
```

whiteout-box

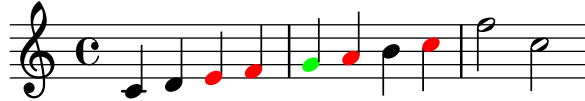
whiteout-rounded-box

whiteout-outline



- 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 |  
}
```



- 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 `midititle` 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).
- 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`.

```
bold-red-markup = \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 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.

- Nella lingua "english" per il nome delle note, i nomi lunghi per le altezze con alterazioni ora contengono un trattino per migliorare la leggibilità. Ora si deve scrivere

`\key a-flat \major`

invece del precedente

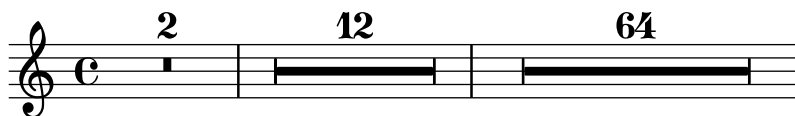
`\key aflat \major`

Le doppie alterazioni non necessitano di un trattino ulteriore, quindi il nome olandese `cisis` corrisponde al nome lungo inglese `c-sharpsharp`.

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



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



- I numeri di pagina ora possono essere stampati in numeri romani impostando la variabile del foglio `page-number-type`.
- È ora possibile usare insieme `\time` e `\partial` per cambiare l'indicazione di tempo a metà misura.

```
\override Score.BarNumber.break-visibility = #end-of-line-invisible
\partial 4 \time 3/4 f4 | 2 4 | 2 \bar "||"
\time 9/8 \partial 4. f8 8 8 | 2. 8 8 8 |
```



- È ora possibile sovrascrivere la proprietà `text` dei nomi degli accordi.

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

A B C<sup>7</sup> bla

- Migliorato l'allineamento orizzontale quando si usa `TextScript`, con `DynamicText` o `LyricText`.
- 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.
- `InstrumentName` ora supporta l'interfaccia `text-interface`.
- 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 { }
}
```

- Ora è più facile usare dei tipi di carattere 'musicali' alternativi al predefinito Emmentaler in LilyPond. Visitare <http://fonts.openlilylib.org/> per maggiori informazioni.
- 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.
- Sono stati introdotti dei miglioramenti al comando `\partial` per evitare i problemi che sorgevano quando si usano molteplici contesti paralleli.



- `\chordmode` può ora usare i costrutti `< >` e `<< >>`.
- 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.

- La funzione `\addlyrics` ora funziona con contesti arbitrari incluso `Staff`.
- 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
```



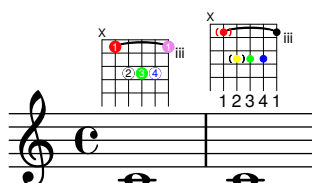
- La proprietà `thin-kern` del grob `BarLine` è stata rinominata `segno-kern`.
- I grob `KeyCancellation` ora ignorano le chiavi delle notine (come fanno anche i grob `KeySignature`).
- Aggiunto il supporto per `\once` `\unset`
- È ora possibile colorare individualmente sia i punti che le parentesi nei 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))
  }
}
}

```

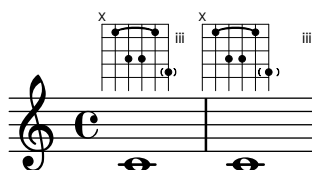


- Sono state aggiunte due nuove proprietà da usare in `fret-diagram-details` quando sia usa il comando `\fret-diagram-verbose` in un blocco markup; `fret-label-horizontal-offset`, che agisce su `fret-label-indication`, e `paren-padding` che regola lo spazio tra il punto e le parentesi che lo circondano.

```

\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))
      }
    }
  }
}

```



- 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

      oooooooooo  oooooooooo      oo      ooo

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

      oooooo      oooooo      oooooo      oooooo

      oooooooooo      oooooooooo      oo      ooo

```

- 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
    }
  }
>>

```



- 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 righe. 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*.
- Il posizionamento dei numeri dei gruppi irregolari per le 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. Ora sono posizionati più vicino alla travatura angolare se esiste un segmento di travatura appropriato per il suo posizionamento e se la parentesi è omessa.  
È stato aggiunto anche il rilevamento delle collisioni, 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. Se il numero è troppo grande per entrare nello spazio disponibile, viene usato il sistema di posizionamento originale basato

sulla parentesi; e in caso di collisione (per esempio con un'alterazione) il numero del gruppo irregolare viene invece spostato verticalmente.

```
\time 3/4
\override Beam.auto-knee-gap = 3
\tuplet 3/2 4 {
  g8 c'' e,
  c'8 g,, e''
  g,,8 e''' c,,
}
```



Il comportamento originale può essere ottenuto attraverso un `\override` e una nuova proprietà `knee-to-beam`.

```
\time 3/4
\override Beam.auto-knee-gap = 3
\override TupletNumber.knee-to-beam = ##f
\tuplet 3/2 4 {
  g8 c'' e,
  c'8 g,, e''
  g,,8 e''' c,,
}
```



- `\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`).
- 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.
- Migliorata la spaziatura verticale delle teste, di dimensione piccola e normale, della nota ‘MI’ negli stili Funk and 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*.
- 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`.

- Le proprietà di contesto nominate nella proprietà `'alternativeRestores'` sono ripristinate al loro valore presente all'inizio della *prima* alternativa in tutte le alternative successive.

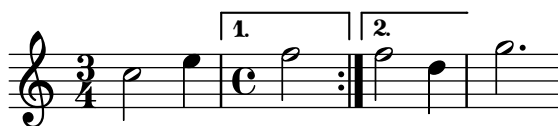
Attualmente l'impostazione predefinita ripristina il 'metro corrente';

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



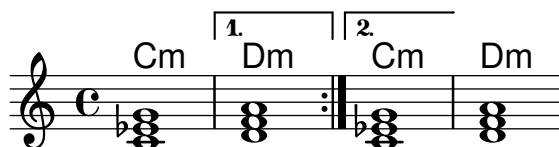
'la posizione della misura';

```
\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 }
}
>>
```



- 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 stata aggiunto un nuovo stile per le teste di nota dell'intavolatura; `TabNoteHead.style = #'slash`.
- Sono stati aggiunti quattro nuovi glifi di chiave: *Doppio Sol*, *Sol tenore*, *varpercussion* e *varDo* e la loro relativa tessitura.

```
\override Staff.Clef.full-size-change = ##t
```

```
\clef "GG" c c c c
\clef "tenorG" c c c c
\clef "varC" c c c c
\clef "altovarC" c c c c
\clef "tenorvarC" c c c c
\clef "baritonevarC" c c c c
\clef "varpercussion" c c c c
```

```
\break
```

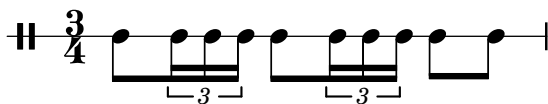
```
\override Staff.Clef.full-size-change = ##f
```

```
\clef "GG" c c c c
\clef "tenorG" c c c c
\clef "varC" c c c c
\clef "altovarC" c c c c
\clef "tenorvarC" c c c c
\clef "baritonevarC" c c c c
\clef "varpercussion" c c c c
```



- Le durate isolate di una sequenza musicale ora sono interpretate come note prive di altezza. Possono essere utili per creare dei ritmi musicali o delle funzioni scheme. Quando sono elaborate per generare la partitura finale, le altezze sono prese dalla nota o accordo precedenti. Ecco due esempi Isolated durations in music sequences now stand for unpitched notes. This may be useful for specifying rhythms to music or scheme functions. When encountered in the final score, the pitches are provided by the preceding note or chord. Here are two che illustrano come l'input sia più facile da leggere:

```
\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 |
}
```



```
\new Staff { r16 c'16 ~ 8 ~ 4 ~ 2 | }
```



- `\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 }
```

- Le eccezioni della disposizione delle travature possono essere costruite con la funzione `scheme \beamExceptions`. Ora si può scrivere

```
\time #'(2 1) 3/16
\set Timing.beamExceptions =
  \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }
c16 c c |
\repeat unfold 6 { c32 } |
```



separando le eccezioni con i segni di controllo di battuta | (scrivere lo schema ritmico senza altezza è comodo ma non obbligatorio). In precedenza le eccezioni dovevano essere specificate così

```
\set Timing.beamExceptions =
#'(
  (end .                               ;inizio della lista associativa
    (                                   ;elemento per la chiusura delle travature
      ((1 . 32) . (2 2 2))             ;inizio della lista associativa per le estremità
      ((1 . 32) . (2 2 2))             ;regola per le travature di 1/32 -- chiudi ogni 1/16
    )
  )
)
```

- 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`.

- 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.