`\extractMusic`

1)

The `\extractMusic` function extracts a fragment of a music expression. You just have to specify the beginning of this fragment, and his length.

The syntax is :
`\extractMusic music from during`

where `music`, `from` and `during` are three music expressions.

`from` and `during` are typically a `\skip` expression. The length of `from` music will give us the position for the first event to extract, relative to the beginning of `music`.The length of the extracted fragment is given by the length of `during`.
For example:
`\extractMusic \music s1 s1*2`
will only keep all events beginning after the first measure (i.e., at the second measure)
but during two measures, so up to the third measure.

2)

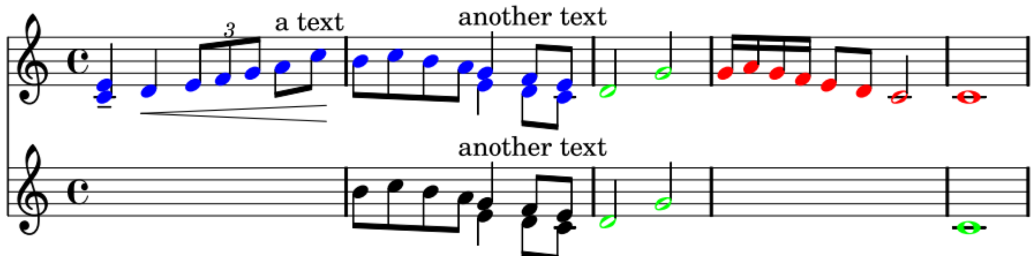
All data (notes, scripts, dynamics and various `\overrides`) outside the range defined by

`from – during` are skipped.

In the above example , the second staff is defined has follow :

```
{ s1 | \extractMusic \music s1 s1*2 | s1 | c'1 }
```

example 1



You can see that the first notes of the second staff are black (not blue) and that the last c'1 is green.

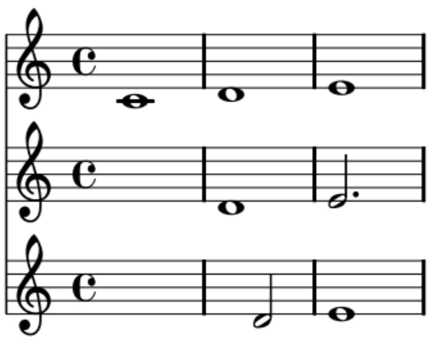
3)

If a note (or an event with a `duration` property) overlaps with one of the two bounds, it is shortened.

In the second staff of the example 2, we have `\extractMusic \music s1 {s1 s2.};`

in the third staff : `\extractMusic \music {s1 s2} { s2 s1}.`

example 2



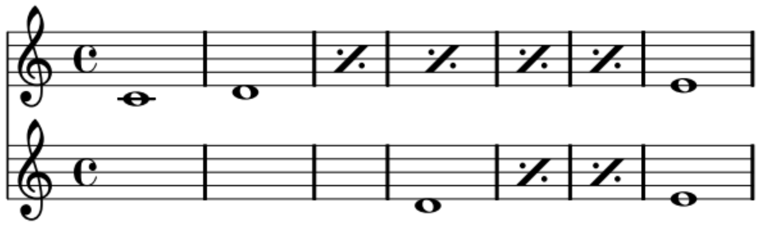
Note that we have added some `\skips` after and before the `\extractMusic` to be clearer.

4)

Musics built with `\repeat` can also be extracted. The repeat-count is adjusted accordingly.

In the second staff, we have `\extractMusic \music s1*3 s1*4`

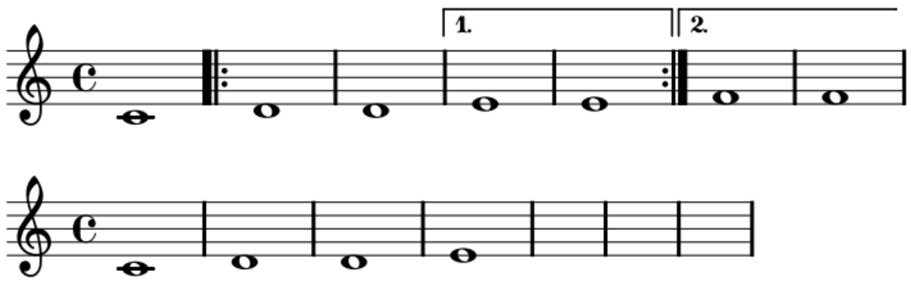
example 3



Sections of music built with `\repeat volta` lose their repeat-structure if one or both bounds of the extraction are inside the structure.

```
 \extractMusic \music   s1*0 s1*4
```

example 4



5)

`\upToMeasure`

In some music with a lot time signature changes, it can be painful and long to compute some `\skips` that fit to the `from` and `during` parameters.

`\upToMeasure` can be helpful in the `from` and `during` parameters of `\extractMusic`.The syntax is :

`\upToMeasure #measure-number` where measure-number is an integer

This function requires first that all time-signature informations are set in a variable called `global.` If no `\global` is found, a time signature of `4/4` is assumed.

Basically, the `\upToMeasure` function results in a `\skip` event with the same duration of a music that would begin measure `1` to the measure `measure-number`, according to the time-signature changes found in `\global`. But, if you use this function inside the `during` parameter of the `\extractMusic` function, `\extractMusic` will detect that, and will adjust the length of this duration, by substracting to it, the length of the `from` parameter.

So, once you have define `\global`, if you write that :

```
\extractMusic \music \upToMeasure #2 {\upToMeasure #5 s2} ,
```

you'll get the music between measure 2 to measure 5, second beat (second beat included).

This feature is used in example 5 below.

Note 1 :

`\extractMusic` doesn't read the `Score.currentBarNumber`. If you want to change it (`\set Score.currentBarNumber = #50` for example), you'd better do this after all your music is completed, to avoid confusions with bars numbering.

Note 2 :

`\upToMeasure` should return the good duration, even if some manual timing tweaks as `\partial` and `\set Timing.measurePosition` are done in `\global`

6)

`\mmR` and `\mmS`

This two functions are intended to be used in the `music` parameter of `\extractMusic`. There are just respectively a `MultiMeasureRest` and a `Skip` with a (quasi) infinite duration.

Writing that, for example :

```
\extractMusic \mmR \upToMeasure #3 \upToMeasure #45
```

will fill your music with `MultiMeasureRest`, whatever the complexity of the time-signature changes found in `\global`.

Please, just remember that a `MultiMeasureRest` cannot be splitted inside a measure.

`\mmS` has not this limitation as shown in the example 5 below.

7)

`\extractBegin` and `\extractEnd`

Their syntaxes are :

```
\extractBegin music during
\extractEnd music from
```

→ `\extractBegin music during` is just a shortcut for :

`\extractMusic music s1*0 during` : extract a fragment from `music`, with the same length than `during`, from the *beginning* of `music`.

→ `\extractEnd music from` extract a fragment from `music`, at the position equal to the length of `from`, up to the *end* of `music`.

The missing `during` parameter is computed automatically from the length of `music`.

If `music` is `\mmR` or `\mmS`, [see 6)], the implicit `during` parameter is computed from the length of `\global`. If no `\global` variable is found, a warning message is written (an infinite number of measures, even empty or filled with `MultiMeasureRest` events, will lead to an infinite time of compilation …). Then, in this case, `\extractEnd` will do nothing.

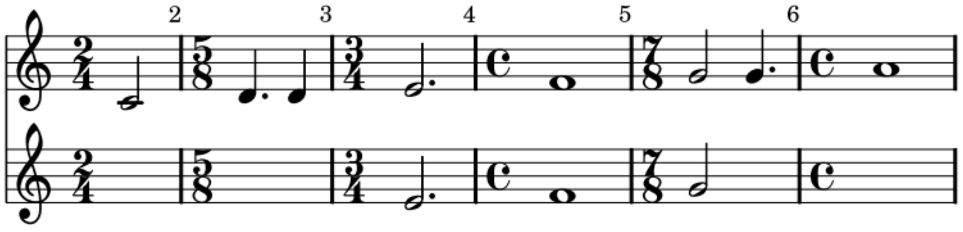
In the example 5, we have :

```
global = { \time 2/4 s2 \time 5/8 s8*5 \time 3/4 s2.
           \time 4/4 s1 \time 7/8 s8*7 \time 4/4 s1 }
```

and the second staff defined as follow

```
{
      \extractBegin \mmS \upToMeasure #3
      \extractMusic \music \upToMeasure #3  { \upToMeasure #5 s2 }
      \extractEnd \mmS   { \upToMeasure #5 s2  }
}
```

example 5



8)

`\insertMusic` and `\replaceMusic`

These two functions are derived from `\extractMusic`

syntax : `\insertMusic music where musicToInsert`

insert `musicToInsert` in `music`, at the position defined by `where`.

Can be useful, for example, to insert some `\override`s in the middle of a music.

syntax : `\replaceMusic music where replacementMusic`

replace with `replacementMusic`, the fragment in `music`, beginning at `where` and during a length equal to the `replacementMusic` length.

`\replaceMusic` works for all kind of music but `VoltaRepeatedMusic`s, because it breaks the repeat-structure.

If you do want to replace a section inside a `\repeat volta … \alternate`, please use the `\replaceVoltaMusic`, which has the same syntax but which re-builds the repeat-structure.

[Well, personally, I prefer to put my repeat-structures only in the `\global` variable, which shrugs off the problem].

\multiExtractMusic

The `\multiExtractMusic` function can speed up a sequence of several `\extractMusic`

For example, if you have :

```
\extractMusic \musicA s1*0 s1*2
\extractMusic \musicB s1*2 s1*3
\extractMusic \musicC s1*5 s1*2
\extractMusic \musicD s1*7 s1*3
```

With that new function, it will be written like that :

```
\multiExtractMusic s1*0 {
     \musicA s1*2
     \musicB s1*3
     \musicC s1*2
     \musicD s1*3
}
```

You do not need anymore to specify the `from` parameter of each `\extractMusic` because they are internally computed by adding the precedent length of the prev fragment.

The syntax is :

```
\multiExtractMusic from seq-music
```

The serie of extractions begins after a length defined by the length of the `from` music.

`seq-music` must be a sequential music of the following form :

```
{  \musicA  \duringA
    \musicB  \duringB
     etc...}
```

The result will be a sequence of this form `{ resultA resultB` etc...`}` with

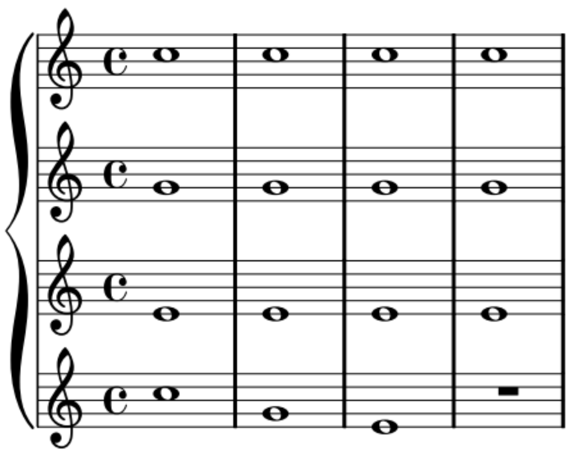
```
resultA=\extractMusic \musicA \from                     \duringA
resultB=\extractMusic \musicB {\from \duringA}          \duringB
resultC=\extractMusic \musicC {\from \duringA \duringB}\duringC
```

etc ...

In this example, the last staff is defined as follow :

```
\multiExtractMusic s1*0 {
  \musicA s1
  \musicB s1
  \musicC s1
  \mmR    s1
}
```

example 6



The use of `\upToMeasure` is of course allowed in the `\duringA`,`\duringB` … parameters of the `seq-music` parameter, but `\multiExtractMusic` checks that the `measure-number` parameter of `\upToMeasure` doesn't lead to a negative `during` length.

```
\multiExtractMusic s1*0 {
     \musicA \upToMeasure #5
     \musicB \upToMeasure #3 % this line is ignored : 3 < 5
}
```

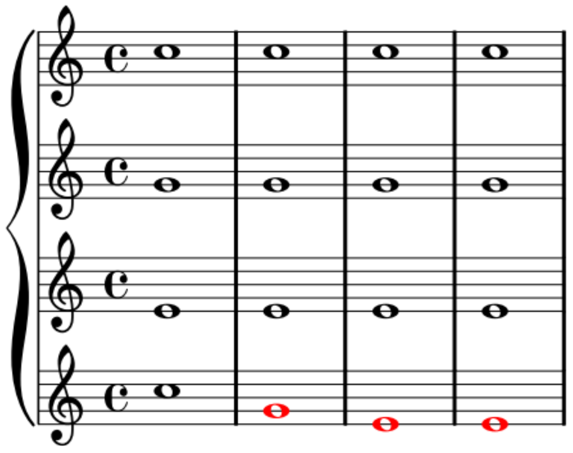
Two little goodies for `\multiExtractMusic`

– In the `seq-music` parameter, the last `during` element can be omitted. If so, the last `music` element is extracted from the current position up to the end of `music` (like would do `\extractEnd`).

– In the `seq-music` parameter, setting a `during` element to a 0 length music (for example s1*0), insert the `music` element untouched (i.e not extract). If this `music` element is `\mmR` or `\mmS`, nothing happens.

Let's see this 2 features in action, with the example 7, identical than example 6 but with now an other last staff definition :

```
\multiExtractMusic s1*0 {
  \musicA              s1
  \override Voice.NoteHead
     #'color = #red    s1*0
  \musicB              s1
  \musicC
}
```



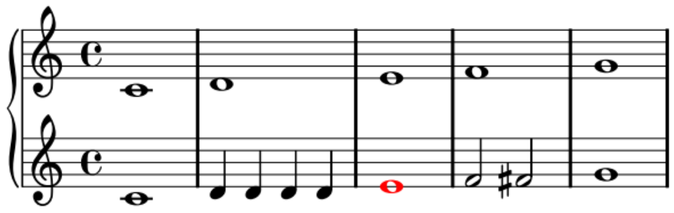
example 7

\multiReplaceMusic

Like `\multiExtractMusic, \multiReplaceMusic` can speed up a sequence of several `\replaceMusic, now.`

Here, the last Staff is just defined as follow :

```
\multiReplaceMusic \music {
  { d4 d d d}              s1
  \once \override Voice.NoteHead
     #'color = #red        s1*2
  fis2                     {s1*3 s2}
}
```



example 8

The `\override` in the third measure has a 0 length, so it is just inserted. Nothing is replaced.

The `fis` in the fourth measure is inserted in the third beat, so it cut the `f1` which become `f2.`

Note that `\multiReplaceMusic` use internally `\replaceMusic` and if you try to replace some music in the middle of a `\repeat volta` structure, this structure will disapear. [The solution is to put `\repeat volta` structures in a `\global` variable]

A few short-cuts

We, here, just make a copy-past of the last lines of "extractMusic.ly" :

```
#(define eM extractMusic)          #(define iM insertMusic)
#(define M upToMeasure)            #(define rM replaceMusic)
#(define eB extractBegin)          #(define mEM multiExtractMusic)
#(define eE extractEnd)            #(define mRM multiReplaceMusic)
```