MusicFormats maintainance guide

https://github.com/jacques-menu/musicformats

v0.9.70 – November 6, 2023

Jacques Menu

This document presents the internal working of MusicFormats. It is part of the MusicFormats documentation, to be found at https://github.com/jacques-menu/musicformats/tree/dev/documentation.

```
void msrSegment::browseData (basevisitor* v)
{
  // ... ... ...

  for (
    std::list<S_msrMeasure>::const_iterator i = fSegmentElementsList.begin ();
    i != fSegmentElementsList.end ();
    ++i
  ) {
    // browse the element
    msrBrowser<msrMeasure> browser (v);
    browser.browse (*(*i));
  } // for

  // ... ... ...
}
```

```
void msr2msrTranslator::visitStart (S_msrClef& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrClef" <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fCurrentVoiceClone->
    appendClefToVoice (elt);
}
```

List of Figures

Listings

Contents

Part I

Overview of MusicFormats

Chapter 1

Introduction

This document presents the design principles and architecture of MusicFormats, as well as information needed to maintain it. It is part of the MusicFormats documentation, to be found at `https://github.com/jacques-menu/musicformats/tree/master/doc`.

All the MusicXML examples mentioned can be downloaded from `https://github.com/jacques-menu/musicformats/tree/master/musicxml`.
They are grouped by subject in subdirectories, such as `basic/HelloWorld.xml`.

The MSDL examples can be found at `https://github.com/jacques-menu/musicformats/tree/master/msdl`.

1.1 Acknowledgements

Many thanks to Dominique Dominique Fober, the designer and maintainer of the `libmusicxml2` library. His two-phase visitor pattern has proved very strong and useful in the implementation of MusicFormats.

Lars Opfermann of the soundnotation company (https://soundnotation.com) has also played an important role, testing `xml2ly` intensively and supplying many MusicXML examples to help fix bugs. This help has been extremely useful and appreciated.

1.2 Prerequisites

In order to maintain MusicFormats, one needs to do the following:

- obtain a working knowledge of C++ programming. The code base of MusicFormats uses classes, simple inheritance, and templates;
- study MusicXML if relevant, starting maybe from `IntroductionToMusicXML.pdf`. A deep knowledge of that subject comes with experience;
- study the architecture of MusicFormats, presented in detail at `MusicFormatsFigures.pdf`.

In this document, all paths to files are relative to the MusicFormats source code directory.

1.3 Chronology

Dominique Fober created `libmusicxml2` long before this author had the need for a library to read MusicXML data, in order to convert it to LilyPond. In the picture showing the architecture of MusicFormats in figure **??** [Architecture], page **??**, Dom's work is essentially represented by the MusicXML, MXSRand Guido boxes at the top. He did more than this, of course, to provide `libmusicxml2` to users!

This author's work started with `xml2ly`, initially named `xml2lilypond`, whose goal was to:

- perform as least as well as `musicxml2ly`, provided by LilyPond;
- provide as many options as needed to meet the user's needs.

The `*.cpp` files in `samples` were examples of the use of the library. Among them, `xml2guido` has been used since in various contexts. The diagram in figure **??** [Architecture], page **??**, was created afterwards, and it would then have consisted of only MusicXML, MXSR and Guido, with passes 1, 2 and 3.

When tackling the conversion of MusicXML to LilyPond, this author created MSR as the central internal representation for music score. It is meant to capture the musical contents of score in fine-grain detail, to meet the needs of creating LilyPond code first, and Braille later. The only change made to the existing MXSR format has been to add an input line number to `xmlElement`.

The conversion from MSR to BSR music was two-pass from the beginning, first creating a BSRformat with unlimited line and page lenghs, and then constraining that in a second BSR would take the numbers of cell per line and lines per page into account. This was frozen in autumn 2019 due to the lack of interest from the numerous persons and bodies that this author contacted about `xml2brl`. The current status is the braille output is that the cells per line and lines per page values are ignored.

The creation of MusicXML code from MSR data was then added to close a loop with MusicXML2xml, with the idea that it would make MusicFormats a kind of swiss knife for textual formats of music scores.

Having implemented a number of computer languages in the past, this author was then tempted to design MSDL, which stands for Music Scores Description Language. The word *description* has been prefered to *programming*, because not all musicians have programming skills. The basic aim of MSDL is to provide a musician-oriented way to describe a score that can be converted to various target textual forms.

`src/clisamples/Mikrokosmos3Wandering.cpp` has been written to check that the MSR API was rich enough to go this way. The API was enriched along the way.

Having MSR, LPSR and BSR available, as well as the capability to generate MusicXML, LilyPond Guidoand Braille, made writing a first draft of the MSDL converter, with version number 1.001, rather easy. The initial output target languages were MusicXML, LilyPond, MusicXML and Braille.

This document contains technical information about the internal working of the code added to MusicFormats by this author as their contribution to this great piece of software.

1.4 Zsh vs Bash

Although the shell mentioned in the MusicFormats user guide is Bash, most scripts and shell functions supplied for MusicFormats maintainance are Zsh scripts. This is because of the magical globbing pattern qualifier `**/` Zsh supplies, which makes `find` seldom needed.

For example, in `devtools/ZshDefinitionsForMusicFormats.zsh`, adding the `include/` folder alongside `src/` is done this way:

```
function addInclude ()
{
  set -x
  echo "--> INCLUDE_DIR: ${INCLUDE_DIR}"

  rm -rf ${INCLUDE_DIR}
  mkdir -p ${INCLUDE_DIR}

  cd ${SRC_DIR}

  rsync -R **/*.h ${INCLUDE_DIR_NAME}

  mv ${INCLUDE_DIR_NAME} ..

  git add ../${INCLUDE_DIR_NAME}/*
  set +x
}
```

This creates the same folders hierarchy as the one in `src/`, with only the `*.h` header files in it.

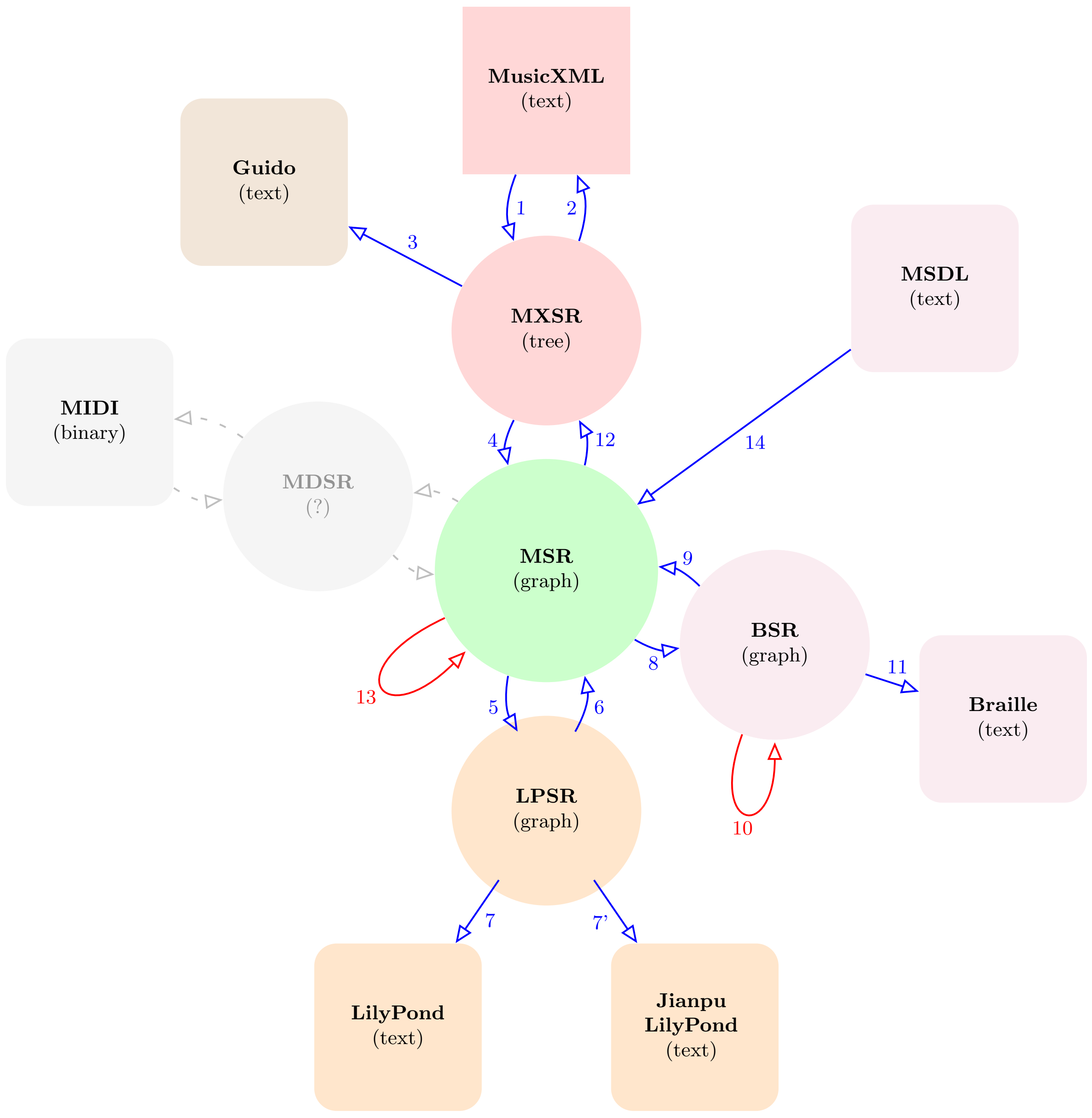
Another useful tool is `devtools/CheckGIndenterUsage.cpp`, whose name is self-comprehensive.

1.5 The GitHub repository

The GitHub repository, hosted at `https://github.com/jacques-menu/musicformats`, contains two branches types:

- the defaultmaster version, to be found at `https://github.com/jacques-menu/musicformats`, is where changes are pushed by the maintainers of MusicFormats. It is the most up to date;
- the `v....` versions are the master versions frozen at some point in time.

Figure 1.1: The MusicFormats architecture



Chapter 2

Documentation

The MusicFormats documentation is written in LaTeX, the pictures being created with the TikZ/PGF package, see https://www.bu.edu/math/2013/08/tikzpgfmanual.pdf.

All the LaTeX files have an initial '! TEX root' comment. This is a TeXShop specific feature, allowing a documentation to be composed from any of the files that it imports, such as:

```
% !TEX root = mfmaintainanceguide.tex
```

The documentation/ directory contains:

```
jacquesmenu@macmini: ~/musicformats-git-dev/documentation > ls -sal
total 48
 0 drwxr-xr-x@  10 jacquesmenu  staff    320 Feb 28 07:42 .
 0 drwxr-xr-x   38 jacquesmenu  staff   1216 Feb 27 12:14 ..
48 -rw-r--r--@   1 jacquesmenu  staff  22532 Feb 28 07:49 .DS_Store
 0 drwxr-xr-x   18 jacquesmenu  staff    576 Feb 28 08:27 CommonLaTeXFiles
 0 drwxr-xr-x   38 jacquesmenu  staff   1216 Feb 18 08:39 IntroductionToMusicXML
 0 drwxr-xr-x   57 jacquesmenu  staff   1824 Feb 28 10:08 mfapiguide
 0 drwxr-xr-x  116 jacquesmenu  staff   3712 Feb 28 10:22 mfmaintainanceguide
 0 drwxr-xr-x   53 jacquesmenu  staff   1696 Feb 28 10:07 mfuserguide
 0 drwxr-xr-x   27 jacquesmenu  staff    864 Feb 14 08:54 graphics
 0 drwxr-xr-x    5 jacquesmenu  staff    160 Jan 23 16:33 presentation
```

The CommonLaTeXFiles/ directory contains LaTeX settings used by the various documentation files and the code for pictures:

```
jacquesmenu@macmini:~/musicformats-git-dev/documentation/CommonLaTeXFiles > ls -sal *.tex
 8 -rwxr--r--@  1 jacquesmenu  staff    241 Aug 17 14:21 CreateMSRScoreRepresentation.zsh
 8 -rw-r--r--   1 jacquesmenu  staff    507 Jun 28 20:52 LaTeXBoxes.tex
16 -rw-r--r--   1 jacquesmenu  staff   6494 Jun 28 20:52 LaTeXCommonSettings.tex
 8 -rw-r--r--   1 jacquesmenu  staff   1907 Jun 28 20:52 LaTeXDivisionsCommands.tex
 8 -rw-r--r--   1 jacquesmenu  staff    957 Jun 28 20:52 LaTeXFontsAndColors.tex
 8 -rw-r--r--   1 jacquesmenu  staff    604 Jun 28 20:52 LaTeXGraphicsAndPictures.tex
 8 -rw-r--r--   1 jacquesmenu  staff   1128 Jun 28 20:52 LaTeXIndexing.tex
24 -rw-r--r--   1 jacquesmenu  staff  10728 Jun 28 20:52 LaTeXListings.tex
 8 -rw-r--r--   1 jacquesmenu  staff   1527 Jun 28 20:52 LaTeXMusicFormatsCommands.tex
24 -rw-r--r--   1 jacquesmenu  staff  11735 Aug 14 16:50 LaTeXMusicFormatsFilesAndFolders.
    tex
 8 -rw-r--r--@  1 jacquesmenu  staff   2151 Jun 28 20:52 LaTeXMusicFormatsNames.tex
 8 -rw-r--r--   1 jacquesmenu  staff    441 Jun 28 20:52 LaTeXMusicNotation.tex
 8 -rw-r--r--   1 jacquesmenu  staff   1535 Jun 28 20:52 LaTeXReferencing.tex
32 -rw-r--r--   1 jacquesmenu  staff  15145 Aug 14 16:50 LaTeXShortcuts.tex
 8 -rw-r--r--   1 jacquesmenu  staff   2171 Jun 28 20:52 LaTeXTablesAndLists.tex
40 -rw-r--r--@  1 jacquesmenu  staff  17746 Jun 28 20:52 MSRClassesHierarchyPicture.tex
40 -rw-r--r--@  1 jacquesmenu  staff  16711 Aug 17 14:21 MSRScoreRepresentation.tex
 8 -rw-r--r--   1 jacquesmenu  staff    321 Jun 28 20:52 MusicFormats.ist
```

```
48 -rw-r--r--   1 jacquesmenu  staff  21749 Jun 28 20:52 MusicFormatsArchitecturePicture.
    tex
 8 -rw-r--r--   1 jacquesmenu  staff   1558 Aug 15 23:36 OAHClassesHierarchyPicture.log
16 -rw-r--r--@  1 jacquesmenu  staff   5906 Aug 16 08:02 OAHClassesHierarchyPicture.tex
```

It is to be noted that documentation/CommonLaTeXFiles/MSRScoreRepresentation.tex is generated from documentation/CommonLaTeXFiles/MSRClassesHierarchyPicture.tex with this script, which should be run only if there is any change in the MSR classes hierarchy:

```
jacquesmenu@macmini-1:~/musicformats-git-dev/documentation/CommonLaTeXFiles > cat
    CreateMSRScoreRepresentation.zsh
#!/bin/zsh

# create a LaTeX file for the user guide from the maintainance one

sed 's/msr//g' MSRClassesHierarchyPicture.tex \
  | \
sed 's/The MSR classes hierarchy/The MSR music score representation/g' \
  > \
MSRScoreRepresentation.tex
```

Directory `graphics/` contains PNG files showing screenshots of the results of using the MusicFormats tools.

Directory `libmusicxml2Presentation/` contains a presentation of `libmusicxml2` written by Dominique Fober.

Directory `IntroductionToMusicxml/` contains a presentation done by this author at the 'MUSIC ENGRAVING IN THE 21ST CENTURY – DEVELOPMENTS AND PERSPECTIVES' conference at Mozarteum in Salzburg in January 2020 (https://www.uni-mozarteum.at/en/kunst/music-engraving-conference.php).

LaTeX needs to be run *three* times when the chapter/section/subsection hierarchy is modified. Check that the last page number, at the bottom of any page, is not less than the one before.

The following files contain the current MusicFormats version number and date:

- the src/MusicFormatsVersionNumber.h and src/MusicFormatsVersionDate.h files are used by the C++ code base;
- file `MusicFormatsVersionNumber.txt` and file `MusicFormatsVersionDate.txt` are used by the LaTeX source files

Those files should be re-generated when a new version of MusicFormats is created, for example:

```
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/SetMusicFormatsVersionNumber.bash "
    0.9.61"
==> PWD is:
/Users/jacquesmenu/musicformats-git-dev

==> Writing MusicFormats version number 0.9.61 to MusicFormatsVersionNumber.txt

8 -rw-r--r--@ 1 jacquesmenu  staff  6 Mar  2 13:43:04 2022 MusicFormatsVersionNumber.txt
0.9.61
==> PWD is:
/Users/jacquesmenu/musicformats-git-dev/src

==> Writing MusicFormats version number 0.9.61 to MusicFormatsVersionNumber.h

8 -rw-r--r--@ 1 jacquesmenu  staff  45 Mar  2 13:43:04 2022 MusicFormatsVersionNumber.h
#define MUSICFORMATS_VERSION_NUMBER "0.9.61"
```

and:

```
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/SetMusicFormatsVersionDate.bash "
    March 2, 2022"
==> PWD is:
/Users/jacquesmenu/musicformats-git-dev

==> Writing MusicFormats version date March 2, 2022 to MusicFormatsVersionDate.txt

8 -rw-r--r--@ 1 jacquesmenu  staff  14 Mar  2 13:43:32 2022 MusicFormatsVersionDate.txt
March 2, 2022

==> PWD is:
/Users/jacquesmenu/musicformats-git-dev/src

==> Writing MusicFormats version date March 2, 2022 to MusicFormatsVersionDate.h

8 -rw-r--r--@ 1 jacquesmenu  staff  50 Mar  2 13:43:32 2022 MusicFormatsVersionDate.h
#define MUSICFORMATS_VERSION_DATE "March 2, 2022"
```

Avoid editing these files manually. In particular, `MusicFormatsVersionNumber.txt` should **NOT** be terminated by an end of line, since its contents is used in the name of library files generated in `build/lib`.

2.1 LATEX macros

The MusicFormats documentation uses a number of macros both to simplify formatting of frequent texts and to feed the many indexes at the end. All of them are grouped in `documentation/CommonLaTeXFiles`:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/documentation/CommonLaTeXFiles >
    ls -sal LaTeX*.tex
 8 -rw-r--r--  1 jacquesmenu  staff    507  Jun 28  20:52  LaTeXBoxes.tex
16 -rw-r--r--  1 jacquesmenu  staff   6494  Jun 28  20:52  LaTeXCommonSettings.tex
 8 -rw-r--r--  1 jacquesmenu  staff   1907  Jun 28  20:52  LaTeXDivisionsCommands.tex
 8 -rw-r--r--  1 jacquesmenu  staff    957  Jun 28  20:52  LaTeXFontsAndColors.tex
 8 -rw-r--r--  1 jacquesmenu  staff    604  Jun 28  20:52  LaTeXGraphicsAndPictures.tex
 8 -rw-r--r--  1 jacquesmenu  staff   1128  Jun 28  20:52  LaTeXIndexing.tex
24 -rw-r--r--  1 jacquesmenu  staff  10728  Jun 28  20:52  LaTeXListings.tex
 8 -rw-r--r--  1 jacquesmenu  staff   1527  Jun 28  20:52  LaTeXMusicFormatsCommands.tex
24 -rw-r--r--@ 1 jacquesmenu  staff  11735  Jul 29  09:02  LaTeXMusicFormatsFilesAndFolders.
    tex
 8 -rw-r--r--@ 1 jacquesmenu  staff   2151  Jun 28  20:52  LaTeXMusicFormatsNames.tex
 8 -rw-r--r--  1 jacquesmenu  staff    441  Jun 28  20:52  LaTeXMusicNotation.tex
 8 -rw-r--r--  1 jacquesmenu  staff   1535  Jun 28  20:52  LaTeXReferencing.tex
32 -rw-r--r--@ 1 jacquesmenu  staff  14665  Jun 28  20:52  LaTeXShortcuts.tex
 8 -rw-r--r--  1 jacquesmenu  staff   2171  Jun 28  20:52  LaTeXTablesAndLists.tex
```

For example:

```
\newcommand{\CLI}{command line\index[Main]{command line}}
```

```
\newcommand{\musicXmlMarkup}[1]{%
{\tt <#1/>}\index[Main]{{\tt $<$#1 /$>$}}\index[MusicXML]{{\tt #1 $<$/$>$}}%
}
\newcommand{\musicXmlAttribute}[1]{%
{\tt "#1"}\index[Main]{{\tt $<$#1 /$>$}}\index[MusicXML]{{\tt #1 ""}}%
}
```

```
\newcommand{\Main}[1]{%
#1\index[Main]{#1}%
}
\newcommand{\MainName}[1]{%
\index[Main]{#1}%
}
```

```

\newcommand{\code}[1]{%
{\tt #1}\index[Main]{{\tt #1}}%
}
```

Some command exist in two forms, differing in the capitalization of the first character:

```
\newcommand{\enumType}{enumeration type\index[Main]{enumeration type}}
\newcommand{\EnumType}{Enumeration type\index[Main]{enumeration type}}
```

Some command names are of the form `*Both*`:

```
\newcommand{\fileName}[1]{%
{\tt #1}\index[Main]{{\tt #1}}\index[Files]{{\tt #1}}%
}
\newcommand{\fileNameBoth}[1]{%
{\textcolor{brown}{\tt *#1.h/.cpp}}\index[Main]{#1.h/.cpp@{{tt *#1.h/.cpp}}}\index[Files
   ]{#1.h/.cpp@{{tt *#1.h/.cpp}}}%
}
```

```
\newcommand{\msrToMsr}[1]{%
{\textcolor{brown}{\tt src/passes/msr2msr/#1}}%
}
\newcommand{\msrToMsrBoth}[1]{%
{\textcolor{brown}{\tt src/passes/msr2msr/#1.h/.cpp}}%
}
```

Some command names are of the form `star*`:

```
\newcommand{\methodName}[1]{%
{\tt #1~()}\index[Main]{{\tt #1}~()}\index[MethodsAndFields]{{\tt #1}~()}%
}
\newcommand{\starMethodName}[1]{%
{\tt *#1~()}\index[Main]{#1~()@{\tt *#1}}\index[MethodsAndFields]{*#1~()@{\tt *#1~()}}%
}
```

Some commands have a variant of the form `*Name*` to produce only their arguments, with no additional text:

```
\newcommand{\file}[1]{%
file {\tt #1}\index[Main]{{\tt #1}}\index[Files]{{\tt #1}}%
}
\newcommand{\File}[1]{%
File {\tt #1}\index[Main]{{\tt #1}}\index[Files]{{\tt #1}}%
}

\newcommand{\fileName}[1]{%
{\tt #1}\index[Main]{{\tt #1}}\index[Files]{{\tt #1}}%
}
\newcommand{\fileNameBoth}[1]{%
{\textcolor{brown}{\tt *#1.h/.cpp}}\index[Main]{#1.h/.cpp@{{tt *#1.h/.cpp}}}\index[Files
   ]{#1.h/.cpp@{{tt *#1.h/.cpp}}}%
}
```

Some commands are in the form `*Repr` : the designate the name of a representation, such as:

```
\newcommand{\msrRepr}{MSR\index[Main]{MSR}}
```

2.2 About this document

This document provides cross views of the information needed for MusicFormats maintainance. It is organized in a number of parts:

- the first part provides an overview of the library, together with the concepts is uses;
- then the two-phase visitors pattern, which is central to MusicFormats, is presented;
- the third part is dedicated to the programming style and conventions used throughout the code base;
- the OAH (Options and help), a pervasive feature in MusicFormats, is detailed;
- the fifth part details the formats provided by the library;
- the following parts are dedicated to passes, generators and converters, respectively;
- the ninth part presents the interfaces to the formats, passes and converters;
- the tenth part provides a longitudinal view of the handling of selected music score contents elements, grouped by such elements such as staves, tuplets and harmonies;
- and finally, the last part is dedicated to the implementation of the MSDL language.

2.3 The MusicFormats architecture

2.4 User guide

`documentation/mfuserguide/mfuserguide.pdf` is the usual user guide. It presents the use of MusicFormats with the command line for the time being.

2.5 API guide

`documentation/mfapiguide/mfapiguide.pdf` presents the use of MusicFormats through the APIs. The latter are used internally by the command line services, and can be used from applications at will, such as in a Web site.

2.6 Maintainance guide

`documentation/mfmaintainanceguide/mfmaintainanceguide.pdf` describes the internals of MusicFormats from a maintainer's point of view. It contains a detailed presentation of the various types used, and a part dedicated to selected topics: this is to have a longitudinal view of how various music elements are handled in the various passes.

Chapter 3

Building MusicFormats

In order to build MusicFormats from source on your machine, you need:

- a C++17 compiler;
- the `cmake` tool.

The supported operating systems both to build the library and run the command line services are Linux, Windows and MacOS. Other systems may be fine but have not been tested.

The C++17 language is needed because MusicFormats uses <`std::regex`> and the `auto` keyword. More recent versions should not be a problem.

3.1 Cloning the repository

Commands such as the following can be used to clone the master and version branches, respectively:

```
MUSIC_FORMATS_DIR=${HOME}/musicformats-git-dev
git clone https://github.com/jacques-menu/musicformats.git ${MUSIC_FORMATS_DIR}
cd ${MUSIC_FORMATS_DEV}
```

```
VERSION_BRANCH=v0.9.59
MUSIC_FORMATS_DIR=${HOME}/musicformats-git-${VERSION_BRANCH}
git clone -b ${VERSION_BRANCH} https://github.com/jacques-menu/musicformats.git ${
    MUSIC_FORMATS_DIR}
cd ${MUSIC_FORMATS_DIR}
```

3.2 One-shot partial `make` runs

Some parts of the source code base have to be created by their own `make` file once and for all. This is the case of:

- the constants and classes generated for MXSR from the MusicXML DTD by `libmusicxml/src/elements/templ`
- the MFSL scanner and parser in the `mfsl/` directory, generated by `flex` and `bison` respectively.

3.3 cmake configuration

This configuration is in build/CMakesList.txt.

3.4 Library Makefile

This Makefile is build/Makefile.

3.5 CLI amples Makefile

This Makefile is src/clisamples/Makefile.

3.6 Building MusicFormats in practise

Once in the local repository clone, just execute:

```
cd build
make
```

The resulting executables are in build/bin:

```
jacquesmenu@macmini: ~/musicformats-git-dev > ll build/bin
total 754368
    0 drwxr-xr-x@ 26 jacquesmenu  staff       832 Sep 27 00:05:02 2021 ./
    0 drwxr-xr-x  11 jacquesmenu  staff       352 Aug  1 18:32:54 2021 ../
72072 -rwxr-xr-x   1 jacquesmenu  staff  36899440 Sep 27 00:04:52 2021 LilyPondIssue34*
72080 -rwxr-xr-x   1 jacquesmenu  staff  36902528 Sep 27 00:04:54 2021
    Mikrokosmos3Wandering*
 8504 -rwxr-xr-x   1 jacquesmenu  staff   4350480 Sep 27 00:04:49 2021 MusicAndHarmonies*
 8504 -rwxr-xr-x   1 jacquesmenu  staff   4350464 Sep 27 00:05:00 2021 RandomChords*
 8504 -rwxr-xr-x   1 jacquesmenu  staff   4350448 Sep 27 00:05:01 2021 RandomMusic*
 8696 -rwxr-xr-x   1 jacquesmenu  staff   4450928 Sep 27 00:04:56 2021 countnotes*
63904 -rwxr-xr-x   1 jacquesmenu  staff  32717248 Sep 27 00:04:57 2021
    libMultipleInitsTest*
76696 -rwxr-xr-x   1 jacquesmenu  staff  39266928 Sep 27 00:05:01 2021 msdlconverter*
  144 -rwxr-xr-x   1 jacquesmenu  staff     70480 Sep 27 00:04:55 2021 musicformatsversion
    *
12616 -rwxr-xr-x   1 jacquesmenu  staff   6455376 Sep 27 00:04:59 2021 partsummary*
 8920 -rwxr-xr-x   1 jacquesmenu  staff   4564864 Sep 27 00:04:59 2021 readunrolled*
81048 -rwxr-xr-x   1 jacquesmenu  staff  41496208 Sep 27 00:04:49 2021 xml2Any*
61232 -rwxr-xr-x   1 jacquesmenu  staff  31347456 Sep 27 00:04:53 2021 xml2brl*
63704 -rwxr-xr-x   1 jacquesmenu  staff  32615072 Sep 27 00:04:47 2021 xml2gmn*
17368 -rwxr-xr-x   1 jacquesmenu  staff   8891744 Sep 27 00:04:56 2021 xml2guido*
63896 -rwxr-xr-x   1 jacquesmenu  staff  32713936 Sep 27 00:04:50 2021 xml2ly*
12512 -rwxr-xr-x   1 jacquesmenu  staff   6403968 Sep 27 00:04:55 2021 xml2midi*
56384 -rwxr-xr-x   1 jacquesmenu  staff  28865024 Sep 27 00:04:59 2021 xml2xml*
 9176 -rwxr-xr-x   1 jacquesmenu  staff   4695472 Sep 27 00:04:55 2021 xmlclone*
 9320 -rwxr-xr-x   1 jacquesmenu  staff   4771024 Sep 27 00:05:00 2021 xmlfactory*
 8912 -rwxr-xr-x   1 jacquesmenu  staff   4559072 Sep 27 00:04:57 2021 xmliter*
 8752 -rwxr-xr-x   1 jacquesmenu  staff   4478336 Sep 27 00:04:55 2021 xmlread*
12104 -rwxr-xr-x   1 jacquesmenu  staff   6193216 Sep 27 00:04:54 2021 xmltranspose*
 9320 -rwxr-xr-x   1 jacquesmenu  staff   4770128 Sep 27 00:05:02 2021 xmlversion*
```

The resulting librairies are in build/bin, here on MacOS:

```
jacquesmenu@macmini: ~/musicformats-git-dev > ll build/lib
total 1888712
      0 drwxr-xr-x  10 jacquesmenu  staff        320 Sep 27 00:04:46 2021 ./
      0 drwxr-xr-x  11 jacquesmenu  staff        352 Aug  1 18:32:54 2021 ../
 104904 -rwxr-xr-x   1 jacquesmenu  staff   53707712 Sep 27 00:04:46 2021 libmusicxml2.3.2.0.dylib*
      0 lrwxr-xr-x   1 jacquesmenu  staff         24 Sep 27 00:04:45 2021 libmusicxml2.3.dylib@ -> libmusicxml2.3.2.0.dylib
1055040 -rw-r--r--   1 jacquesmenu  staff  532838416 Sep 27 00:04:41 2021 libmusicxml2.a
 591776 -rw-r--r--   1 jacquesmenu  staff  302989312 Sep 21 09:05:55 2021 libmusicxml2.a.A93i4n
  57056 -rw-r--r--   1 jacquesmenu  staff   29212672 Sep 21 09:01:27 2021 libmusicxml2.a.KHrJTO
  39968 -rw-r--r--   1 jacquesmenu  staff   20463616 Sep 21 09:11:20 2021 libmusicxml2.a.gZfmqe
  39968 -rw-r--r--   1 jacquesmenu  staff   20463616 Sep 21 09:09:22 2021 libmusicxml2.a.tndUAV
      0 lrwxr-xr-x   1 jacquesmenu  staff         20 Sep 27 00:04:45 2021 libmusicxml2.dylib@ -> libmusicxml2.3.dylib
```

Chapter 4

Code base structure

The code base of the MusicFormats library contains:

- `build` : a set of files to build the library in various environments with `make`
- `doc` : the documentation in LaTeX source and PDF formats
- `files` : a set of sample files for MusicXML and MSDL
- `javascript` : a set of files for the generation of Java Script, to allow the use of MusicFormats in Web sites
- `libmusicxml` : an embedded copy of the `libmusicxml2` code base
- `packages` : a set of files to create installable versions of the library, not yet operational
- `samples` : the main programs for examples supplide with MusicFormats, such as generators and converters
- `schemas` : a set of files defining the input languages, currently MusicXML, BMML and MEI, together with scripts to generated the set of classes definitions for analyzing them
- `src` the library code base, detailed below
- `validation` : a set of files including a `Makefile` for the validation of the library using the contents of `files`
- `win32` : Windows related support

4.1 The `libmusicxml` folder

This folder contains a version of Grame's `libmusicxml2` library, available at https://github.com/grame-cncm/libmusicxml. It is used by MusicFormats, to avoid the need for installing it separately.

The only possible annoyance when installing both libraries is that the executables in `libmusicxml/build/bin` such as `countnotes` and `xml2guido` are installed twice: choosing which one to use can be handled in the `${PATH}` and `${LD_LIBRARY_PATH}` environment variables or their equivalent.

4.1.1 Embedding libmusicxml in MusicFormats

libmusicxml2 was cloned initally like this:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev > git clone https://github.com/
    grame-cncm/libmusicxml -b dev
Cloning into 'libmusicxml'...
remote: Enumerating objects: 56386, done.
remote: Counting objects: 100% (4692/4692), done.
remote: Compressing objects: 100% (777/777), done.
remote: Total 56386 (delta 3917), reused 4671 (delta 3904), pack-reused 51694
Receiving objects: 100% (56386/56386), 105.98 MiB | 15.91 MiB/s, done.
Resolving deltas: 100% (46834/46834), done.
```

libmusicxml2 is quite stable, and it can be upgraded if needed with:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml > git pull
Already up to date.
```

There is no need to build libmusicxml2 manually, since its code is taken into account by the MusicFormats Makefile.

4.1.2 Upgrading the supported MusicXML version

It may be necessary to upgrade the MusicXML DTD in libmusicxml/elements to keep up with evolutions if libmusicxml2 is not up to date yet.

To upgrade from MusicXML 3.1 to MusicXML 4.0, the following has been done. Makefile_ORIGINAL is a symbolic link to the Makefile provided by libmusicxml2 for 3.1 at the time of this writing:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/schema > ls -sal
total 208
 0 drwxr-xr-x  14 jacquesmenu  staff   448 Jul 30 05:59 .
 0 drwxr-xr-x  22 jacquesmenu  staff   704 Jul 29 07:19 ..
 0 drwxr-xr-x   6 jacquesmenu  staff   192 Jul 29 07:19 2.0
 0 drwxr-xr-x   6 jacquesmenu  staff   192 Jul 29 07:19 3.0
 0 drwxr-xr-x   6 jacquesmenu  staff   192 Jul 29 07:19 3.1
 0 drwxr-xr-x   8 jacquesmenu  staff   256 Jul 29 08:04 4.0
 8 -rw-r--r--   1 jacquesmenu  staff  1215 Jul 29 07:19 Makefile
 8 -rw-r--r--   1 jacquesmenu  staff  1215 Jul 29 07:19 Makefile_3.1
 0 lrwxr-xr-x   1 jacquesmenu  staff     8 Jul 30 05:59 Makefile_ORIGINAL -> Makefile
```

First, create the Makefile for version 4.0:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/schema > sed 's
    /3.1/4.0/g' Makefile_3.1 > Makefile_4.0
```

Then use it to create the C++ files containing the constants and types to be used by MXSR, such as k_accidental, S_accidental:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/schema > make -f
    Makefile_4.0
grep "<xs:element" 4.0/musicxml.xsd | sed -e 's/^.*name="//' | sed -e 's/"..*//' | sort -u
     > elements.txt
../src/elements/templates/elements.bash elements.txt ../src/elements/templates constants >
     elements.h   || rm -f elements.h
../src/elements/templates/elements.bash elements.txt ../src/elements/templates types >
    typedefs.h   || rm -f typedefs.h
../src/elements/templates/elements.bash elements.txt ../src/elements/templates map >
    factory.cpp || rm -f factory.cpp
```

The resulting files are the following, where `elements.txt` contains an alphabetic list of the MusicXML markups found in the DTD:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/schema > ls -sal
total 208
 0 drwxr-xr-x  13 jacquesmenu  staff    416 Jul 29 08:27 .
 0 drwxr-xr-x  22 jacquesmenu  staff    704 Jul 29 07:19 ..
 0 drwxr-xr-x   6 jacquesmenu  staff    192 Jul 29 07:19 2.0
 0 drwxr-xr-x   6 jacquesmenu  staff    192 Jul 29 07:19 3.0
 0 drwxr-xr-x   6 jacquesmenu  staff    192 Jul 29 07:19 3.1
 0 drwxr-xr-x   8 jacquesmenu  staff    256 Jul 29 08:04 4.0
 8 -rw-r--r--   1 jacquesmenu  staff   1215 Jul 29 07:19 Makefile_3.1
 8 -rw-r--r--@  1 jacquesmenu  staff   1215 Jul 29 08:26 Makefile_4.0
 8 -rw-r--r--   1 jacquesmenu  staff   1215 Jul 29 07:19 Makefile_ORIGINAL
16 -rw-r--r--   1 jacquesmenu  staff   7130 Jul 29 08:27 elements.h
16 -rw-r--r--   1 jacquesmenu  staff   4561 Jul 29 08:27 elements.txt
96 -rw-r--r--   1 jacquesmenu  staff  46341 Jul 29 08:27 factory.cpp
56 -rw-r--r--   1 jacquesmenu  staff  24604 Jul 29 08:27 typedefs.h
```

The mapping between the makups text and the types that describes them is done with:

```
typedef SMARTP<musicxml<k_accidental> >   S_accidental;

fMap["accidental"]  = new newElementFunctor<k_accidental>;
```

Finally, copy the new C++ files to the `libmusicxml/elements` folder:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/schema > cp -p
    elements.h factory.cpp typedefs.h ../src/elements
```

Now, rebuild MusicFormats, for it to use the new MusicXML DTD: /libdir/CMakeCache.txt

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/build > rm libdir/CMakeCache.txt
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/build > make
```

It may happen that error messages regarding the new markups are issued:

```
Undefined symbols for architecture x86_64:
  "MusicFormats::mxsr2msrTranslator::visitStart(MusicXML2::SMARTP<MusicXML2::musicxml<241>
     >&)", referenced from:
  "MusicFormats::mxsr2msrTranslator::visitStart(MusicXML2::SMARTP<MusicXML2::musicxml<266>
     >&)", referenced from:
  "MusicFormats::mxsr2msrTranslator::visitStart(MusicXML2::SMARTP<MusicXML2::musicxml<284>
     >&)", referenced from:
  "MusicFormats::mxsr2msrTranslator::visitStart(MusicXML2::SMARTP<MusicXML2::musicxml<29>
    >&)", referenced from:
  "non-virtual thunk to MusicFormats::mxsr2msrTranslator::visitStart(MusicXML2::SMARTP<
    MusicXML2::musicxml<284> >&)", referenced from:
  "non-virtual thunk to MusicFormats::mxsr2msrTranslator::visitStart(MusicXML2::SMARTP<
    MusicXML2::musicxml<29> >&)", referenced from:
clang: error: linker command failed with exit code 1 (use -v to see invocation)
** BUILD FAILED **
make[1]: *** [macos] Error 65
make: *** [all] Error 2
```

In that case, the corresponding constants can be found in `elements.h`, at line '26+numericalValue'. For example, with MusicXML version 4.0, 241 is the numerical value of `k_notations`, describing markup `<"notations"/>`:

```
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/src/elements > grep -
    n kNoElement elements.h
26: kNoElement,
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/src/elements > grep -
    n k_notations elements.h
267:  k_notations,
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/src/elements > grep -
    n k_other_notation elements.h
292:  k_other_notation,
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/src/elements > grep -
    n k_part_name elements.h
310:  k_part_name,
311:  k_part_name_display,
jacquesmenu@mac-mini-de-jacques-1:~/musicformats-git-dev/libmusicxml/src/elements > grep -
    n k_bass elements.h
55: k_bass,
56: k_bass_alter,
57: k_bass_separator,
58: k_bass_step,
```

The incriminated MXSR elements are thus `k_notations`, `k_other_notation`, `k_part_name` and `k_bass`.

The first 4 error messages above mean that the corresponding `visitStart ()` methods are declared alright, but are not defined in `src/passes/mxsr2msr/mxsr2msrTranslator.cpp`.

The following methods definitions are thus missing:

- method `mxsr2msrTranslator::visitStart ( S_notations& elt)`
- method `mxsr2msrTranslator::visitStart ( S_other_notation& elt)`
- method `mxsr2msrTranslator::visitStart ( S_part_name& elt)`
- method `mxsr2msrTranslator::visitStart ( S_bass& elt)`

Type `S_part_name` is there by mistake (some typing was done before the upgrade to 4.0), since it is handled in class `mxsr2msrSkeletonBuilder`, and the other 3 are new in MusicXML 4.0.

4.2 The `doc` folder

This folder contains `LaTeXCommonSettings.tex`, included by the various LaTeX documents whose code is in the respective folders, together with the PDF files:

```
jacquesmenu@macmini:~/musicformats-git-dev/documentation > ll
total 32
 0 drwxr-xr-x   11 jacquesmenu  staff    352 Nov  3 09:59:31 2022 ./
 0 drwxr-xr-x   34 jacquesmenu  staff   1088 Nov  3 16:17:01 2022 ../
32 -rw-r--r--@   1 jacquesmenu  staff  14340 Nov  3 14:15:54 2022 .DS_Store
 0 drwxr-xr-x   25 jacquesmenu  staff    800 Nov  3 09:57:22 2022 CommonLaTeXFiles/
 0 drwxr-xr-x   32 jacquesmenu  staff   1024 Jun 28 20:52:12 2022 IntroductionToMusicXML/
 0 drwxr-xr-x   48 jacquesmenu  staff   1536 Nov  1 14:24:27 2022 mfapiguide/
 0 drwxr-xr-x    9 jacquesmenu  staff    288 Nov  3 09:47:22 2022 MusicFormatsFigures/
 0 drwxr-xr-x  106 jacquesmenu  staff   3392 Nov  3 10:58:57 2022 mfmaintainanceguide/
 0 drwxr-xr-x   55 jacquesmenu  staff   1760 Nov  1 14:24:15 2022 mfuserguide/
 0 drwxr-xr-x   46 jacquesmenu  staff   1472 Aug 31 11:09:53 2022 mfgraphics/mfgraphics
 0 drwxr-xr-x    5 jacquesmenu  staff    160 Jun 28 20:52:12 2022 libmusicxml2Presentation
    /
```

`common` contains a set of files used by the various documents and various stuff:

```
jacquesmenu@macmini: ~/musicformats-git-dev/documentation/common > ll
total 776
  0 drwxr-xr-x   9 jacquesmenu  staff     288 Jan  4 17:23:41 2022 ./
  0 drwxr-xr-x   9 jacquesmenu  staff     288 Jan  4 17:23:41 2022 ../
624 -rw-r--r--   1 jacquesmenu  staff  318497 Apr 22 15:48:40 2021 INSIDE.pdf
  8 -rw-r--r--@  1 jacquesmenu  staff     321 Sep  8 18:15:51 2021 MusicFormats.ist
 48 -rw-r--r--@  1 jacquesmenu  staff   21751 Dec 28 18:45:25 2021
    MusicFormatsArchitecturePicture.tex
 80 -rw-r--r--@  1 jacquesmenu  staff   39133 Jan  4 17:18:28 2022 LaTeXCommonSettings.tex
  8 -rwxr-xr-x@  1 jacquesmenu  staff     157 Jan  4 09:43:30 2022
    createCurrentVersionNumberString.bash*
  0 drwxr-xr-x  12 jacquesmenu  staff     384 Apr 22 15:48:41 2021 images_KEEP/
  8 -rw-r--r--@  1 jacquesmenu  staff       7 Jan  4 09:25:02 2022
    MusicFormatsVersionNumber.txt
```

The `presentation` sub-folder contains the documentation of the `libmusicxml2` library, written by Dominique Fober:

```
jacquesmenu@macmini:~/musicformats-git-dev/documentation/libmusicxml2Presentation > ll
total 416
  0 drwxr-xr-x   5 jacquesmenu  staff     160 Jun 28 20:52:12 2022 ./
  0 drwxr-xr-x  11 jacquesmenu  staff     352 Nov  3 09:59:31 2022 ../
  0 drwxr-xr-x   6 jacquesmenu  staff     192 Jun 28 20:52:12 2022 imgs/
392 -rw-r--r--@  1 jacquesmenu  staff  200524 Jun 28 20:52:12 2022 libmusicxml2.pdf
 24 -rw-r--r--   1 jacquesmenu  staff   11017 Jun 28 20:52:12 2022 libmusicxml2.tex
```

4.3 The `schemas` folder

This folder contains the definitions used to create the classes definitions to analyze textual data in the MusicXML, MEI and BMML formats.

In the `devtools/` folder, `elements.bash` compiles the definitions of MusicXML markups into the C++ code files containing the corresponding C++ classes:

```
jacquesmenu@macmini: ~/musicformats-git-dev/schemas > ll
total 2576
   0 drwxr-xr-x   9 jacquesmenu  staff      288 May 21 18:30:08 2021 ./
   0 drwxr-xr-x  22 jacquesmenu  staff      704 May 25 17:19:16 2021 ../
  16 -rw-r--r--   1 jacquesmenu  staff     6148 May 21 18:30:08 2021 .DS_Store
   0 drwxr-xr-x   4 jacquesmenu  staff      128 Apr 22 15:49:14 2021 BMML/
   0 drwxr-xr-x   5 jacquesmenu  staff      160 May 21 18:30:08 2021 MEI/
   8 -rw-r--r--   1 jacquesmenu  staff     2502 Apr 22 15:49:15 2021 Makefile
   0 drwxr-xr-x   6 jacquesmenu  staff      192 May 21 18:30:08 2021 MusicXML/
2552 -rw-r--r--   1 jacquesmenu  staff  1305905 Apr 22 15:49:13 2021 mei-CMN.rng
   0 drwxr-xr-x   3 jacquesmenu  staff       96 Apr 22 15:49:08 2021 scripts/
```

4.4 The `src` folder

The `src` folder has the following structure:

- `clisamples` : the `main ()` functions of the various command line executables provided by MusicFormats:

```
jacquesmenu@macmini: ~/musicformats-git-dev > ll clisamples/
total 320
 0 drwxr-xr-x  16 jacquesmenu  staff    512 May 24 10:58:19 2021 ./
 0 drwxr-xr-x  22 jacquesmenu  staff    704 May 25 17:19:16 2021 ../
16 -rw-r--r--   1 jacquesmenu  staff   6148 May 21 18:30:07 2021 .DS_Store
 8 -rw-r--r--   1 jacquesmenu  staff    116 Apr 22 15:49:06 2021 .gitignore
40 -rw-r--r--@  1 jacquesmenu  staff  20239 May 24 11:17:46 2021 LilyPondIssue34.cpp
 8 -rw-r--r--   1 jacquesmenu  staff   1615 Apr 22 15:49:15 2021 Makefile
40 -rw-r--r--@  1 jacquesmenu  staff  20239 May 24 11:07:21 2021
   Mikrokosmos3Wandering.cpp
24 -rw-r--r--   1 jacquesmenu  staff   9941 May 21 18:30:07 2021 MusicAndHarmonies.
   cpp
 8 -rw-r--r--   1 jacquesmenu  staff   3114 Apr 22 15:49:15 2021 libMultipleInitsTest
   .cpp
48 -rw-r--r--   1 jacquesmenu  staff  23061 May 21 18:30:07 2021 msdl.cpp
 8 -rw-r--r--   1 jacquesmenu  staff    895 May 21 18:30:07 2021 musicformatsversion.
   cpp
24 -rw-r--r--   1 jacquesmenu  staff  10492 Apr 22 15:49:14 2021 xml2Any.cpp
24 -rw-r--r--   1 jacquesmenu  staff  10076 May 21 18:30:07 2021 xml2brl.cpp
24 -rw-r--r--   1 jacquesmenu  staff  10515 May 21 18:30:07 2021 xml2gmn.cpp
24 -rw-r--r--   1 jacquesmenu  staff  10309 May 21 18:30:07 2021 xml2ly.cpp
24 -rw-r--r--   1 jacquesmenu  staff  10463 May 21 18:30:08 2021 xml2xml.cpp
```

- `converters` : the multi-pass converter combining those in `passes`
  - `msdl2braille`
  - `msdl2guido`
  - `msdl2lilypond`
  - `msdl2musicxml`
  - `msdlconverter`

  - `msr2braille`
  - `msr2guido`
  - `msr2lilypond`
  - `msr2musicxml`

  - `musicxml2braille`
  - `musicxml2guido`
  - `musicxml2lilypond`
  - `musicxml2musicxml`

- `generators` :
  - `LilyPondIssue34`
  - `Mikrokosmos3Wandering`
- `components` : the MusicFormats components formats, including versions numbering and history:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll components
total 168
 0 drwxr-xr-x   7 jacquesmenu  staff    224 Oct 22 08:53:06 2021 ./
 0 drwxr-xr-x  19 jacquesmenu  staff    608 Oct 22 05:29:29 2021 ../
 8 -rw-r--r--@  1 jacquesmenu  staff   1106 Oct 22 09:16:21 2021 mfcComponents.h
96 -rw-r--r--@  1 jacquesmenu  staff  46691 Nov  8 12:59:57 2021 mfcBasicTypes.cpp
40 -rw-r--r--@  1 jacquesmenu  staff  20121 Nov  8 12:59:43 2021 mfcBasicTypes.h
16 -rw-r--r--@  1 jacquesmenu  staff   4950 Nov  8 12:59:08 2021 mfcLibraryComponent.
   cpp
 8 -rw-r--r--@  1 jacquesmenu  staff    605 Oct 22 10:36:30 2021 mfLibraryComponent.h
```

- `mfutilities` : various utilities, including indented text output streams, and version history support:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll mfutilities/
total 200
 0 drwxr-xr-x  15 jacquesmenu  staff    480 Oct 22 06:25:57 2021 ./
 0 drwxr-xr-x  19 jacquesmenu  staff    608 Oct 22 05:29:29 2021 ../
 8 -rw-r--r--@  1 jacquesmenu  staff   3255 Oct 18 20:22:16 2021 mfBool.cpp
16 -rw-r--r--@  1 jacquesmenu  staff   4917 Oct 18 19:56:51 2021 mfBool.h
 8 -rw-r--r--@  1 jacquesmenu  staff   1336 Oct 15 18:48:10 2021 mfEnumAll.h
16 -rw-r--r--@  1 jacquesmenu  staff   7182 Nov  8 13:08:51 2021 mfIndentedTextOutput
    .cpp
16 -rw-r--r--@  1 jacquesmenu  staff   7715 Nov  8 13:08:40 2021 mfIndentedTextOutput
    .h
 8 -rw-r--r--@  1 jacquesmenu  staff    889 Oct 15 20:34:47 2021
    mfMusicformatsErrorKind.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff    629 Oct 15 20:34:47 2021 mfMusicformatsErrors
    .h
 8 -rw-r--r--@  1 jacquesmenu  staff   2541 Nov  5 11:29:25 2021 oahOptionsVector.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff    972 Oct 15 20:16:51 2021 oahBasicTypes.h
64 -rw-r--r--@  1 jacquesmenu  staff  29773 Oct 15 18:48:10 2021 mfStringsHandling.
    cpp
16 -rw-r--r--@  1 jacquesmenu  staff   6269 Oct 15 18:55:46 2021 mfStringsHandling.h
16 -rw-r--r--@  1 jacquesmenu  staff   5028 Oct  7 20:03:27 2021 mfTiming.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff   3726 Oct  8 08:21:09 2021 mfTiming.h
```

- `oah` : object-oriented Options And Help support

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll oah
total 1456
  0 drwxr-xr-x  34 jacquesmenu  staff    1088 Nov 16 08:12:11 2021 ./
  0 drwxr-xr-x  20 jacquesmenu  staff     640 Nov 16 08:12:03 2021 ../
 48 -rw-r--r--@  1 jacquesmenu  staff   23743 Nov 16 08:16:55 2021 basicOah2manPage.
    cpp
 16 -rw-r--r--@  1 jacquesmenu  staff    5202 Nov 15 12:56:16 2021
    oahBasicOah2manPage.h
  8 -rw-r--r--@  1 jacquesmenu  staff     539 Jun  6 06:38:55 2021
    mfEnableHarmoniesExtraSetting.h
  8 -rw-r--r--@  1 jacquesmenu  staff     526 Oct 11 11:56:29 2021
    mfPreprocessorSettings.h
 72 -rw-r--r--@  1 jacquesmenu  staff   34280 Nov 16 08:16:55 2021 harmoniesExtraOah.
    cpp
 24 -rw-r--r--@  1 jacquesmenu  staff   11848 Nov 15 12:56:16 2021
    oahHarmoniesExtraOah.h
 16 -rw-r--r--@  1 jacquesmenu  staff    5154 Nov 16 08:16:55 2021
    harmoniesExtraOah2manPage.cpp
  8 -rw-r--r--@  1 jacquesmenu  staff    1689 Nov 15 12:56:16 2021
    oahHarmoniesExtraOah2manPage.h
  8 -rw-r--r--@  1 jacquesmenu  staff     918 Nov 16 08:16:55 2021 oah2manPage.cpp
  8 -rw-r--r--@  1 jacquesmenu  staff     912 Nov 15 12:56:16 2021 oah2manPage.h
344 -rw-r--r--@  1 jacquesmenu  staff  175094 Nov 16 08:16:55 2021 oahAtomsCollection
    .cpp
176 -rw-r--r--@  1 jacquesmenu  staff   87460 Nov 15 12:56:16 2021 oahAtomsCollection
    .h
336 -rw-r--r--@  1 jacquesmenu  staff  168969 Nov 16 08:16:55 2021 oahBasicTypes.cpp
 96 -rw-r--r--@  1 jacquesmenu  staff   47228 Nov 15 12:56:16 2021 oahBasicTypes.h
  8 -rw-r--r--@  1 jacquesmenu  staff    3258 Nov 16 08:16:55 2021 oahBrowsers.h
 32 -rw-r--r--@  1 jacquesmenu  staff   14030 Nov 16 08:16:55 2021 oahElements.cpp
 24 -rw-r--r--@  1 jacquesmenu  staff   10381 Nov 15 12:56:16 2021 oahElements.h
  8 -rw-r--r--@  1 jacquesmenu  staff    2577 Nov 16 08:16:55 2021 oahInsiderHandlers
    .cpp
  8 -rw-r--r--@  1 jacquesmenu  staff    2982 Nov 15 12:56:16 2021 oahInsiderHandlers
    .h
 56 -rw-r--r--@  1 jacquesmenu  staff   25901 Nov 16 08:16:55 2021 oahOah.cpp
 32 -rw-r--r--@  1 jacquesmenu  staff   13849 Nov 16 08:16:55 2021 oahOah.h
  8 -rw-r--r--@  1 jacquesmenu  staff    1966 Nov 16 08:16:55 2021 oahOah2manPage.cpp
  8 -rw-r--r--@  1 jacquesmenu  staff    1021 Nov 15 12:56:16 2021 oahOah2manPage.h
 24 -rw-r--r--@  1 jacquesmenu  staff    8831 Nov 16 08:16:55 2021 oahRegularHandlers
   .cpp
  8 -rw-r--r--@  1 jacquesmenu  staff    3855 Nov 15 12:56:16 2021 oahRegularHandlers
   .h
  8 -rw-r--r--@  1 jacquesmenu  staff     568 Nov 15 12:56:16 2021 oahVisitor.cpp
  8 -rw-r--r--@  1 jacquesmenu  staff     894 Nov 15 12:56:16 2021 oahVisitor.h
 16 -rw-r--r--@  1 jacquesmenu  staff    5978 Nov 16 08:16:55 2021 outputFileOah.cpp
  8 -rw-r--r--@  1 jacquesmenu  staff    3593 Nov 15 12:56:16 2021 mfOutputFileOah.h
  8 -rwxr--r--@  1 jacquesmenu  staff     236 Oct 23 12:02:12 2021 zsh_test.zsh*
```

- `formatsgeneration` : support for various output kinds
  - `brailleGeneration`
  - `guidoGeneration`
  - `lilypondGeneration`
  - `msrGeneration`
  - `multiGeneration`
  - `mxsrGeneration`
- `passes` : code for the individual passes
  - `bsr2braille`
  - `bsr2bsr`

  - `lpsr2lilypond`

  - `msr2bsr`
  - `msr2lpsr`
  - `msr2msr`
  - `msr2mxsr`

  - `mxsr2guido`
  - `mxsr2msr`
  - `mxsr2musicxml`
- `formats` : the various internal representations used by MusicFormats
  - `bsr`
  - `lpsr`
  - `msdl`
  - `msdr`
  - `msr`
  - `msrapi`
  - `mxsr`
- `wae` : multilingual Warnings And Errors support, including exceptions handling

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll wae/
total 104
 0 drwxr-xr-x   9 jacquesmenu  staff    288 Oct 15 20:23:46 2021 ./
 0 drwxr-xr-x  20 jacquesmenu  staff    640 Nov 16 08:12:03 2021 ../
 8 -rw-r--r--@  1 jacquesmenu  staff    680 Jun  6 06:35:19 2021
    mfEnableAbortToDebugErrorsSetting.h
 8 -rw-r--r--@  1 jacquesmenu  staff    602 Nov 15 12:56:18 2021 waeExceptions.cpp
24 -rw-r--r--@  1 jacquesmenu  staff  11514 Nov 15 12:56:18 2021 waeExceptions.h
 8 -rw-r--r--@  1 jacquesmenu  staff   1393 Nov 16 08:16:55 2021 waeHandlers.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff   1550 Nov 15 12:56:18 2021 waeHandlers.h
32 -rw-r--r--@  1 jacquesmenu  staff  16317 Nov 15 12:56:18 2021 wae.cpp
16 -rw-r--r--@  1 jacquesmenu  staff   5794 Nov 15 12:56:18 2021 waeInterface.h
```

4.5 The validation folder

This folder contains a Makefile to compile all the files in the files folder. musicformatsversion.txt contains a validation version number, without a priori relation to the actual version number of the library, for example:

```
jacquesmenu@macmini: ~/musicformats-git-dev/validation > cat musicformatsversion.txt
1.0.0
```

In this example, make will create a folder named 1.0.0 containing the converted files, including PDF files produced by lilypond.

This validation version number allows for comparisions between version to ease regression tests.

There is no clean target in the Makefile. Removing the 1.0.0 folder in this case does the equivalent, then we can run make again.

Chapter 5

Components

5.1 Components terminology

In compiler writing terminology:

- an external format
- an internal representation is a data structure representing the program being compiled;
- there are often several internal representations, to simplify the compiler internal workings or for optimisation purposes;
- the output of the compiler, such as binary code for some physical or emulated processor, is a last 'representation' of the program;
- a pass converts an internal representation into another one, in a single step;
- a multi-pass converter is a chain of passes, reading the input, converting it into a first internal representation, then a pass to convert it into another internal representation, and so on until the compiler output is produced.

MusicFormats maps exactly to this model, providing the following components:

- internal representations (formats for short) of the music score: MSR, LSPR, BSR and MXSR;
- several passes are available to convert such formats into others;
- a set of multi-pass converters are supplied, such as `xml2ly` `xml2xml` and MSDL converter.

In the MusicFormats user documentation, the term 'converter' is used because it is more meaningfull for musicians.

MusicFormats provides high-level interfaces to its components as functions in `Interface` files:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > look Interface
./formats/msr/msrInterface.cpp
./formats/msr/msrInterface.h
./formats/lpsr/lpsrInterface.cpp
./formats/lpsr/lpsrInterface.h
./formats/bsr/bsrInterface.h
./formats/bsr/bsrInterface.cpp
./passes/mxsr2musicxml/mxsr2musicxmlTranlatorInterface.h
./passes/mxsr2musicxml/mxsr2musicxmlTranlatorInterface.cpp
./passes/bsr2bsr/bsr2bsrFinalizerInterface.h
./passes/bsr2bsr/bsr2bsrFinalizerInterface.cpp
./passes/msr2mxsr/msr2mxsrInterface.cpp
./passes/msr2mxsr/msr2mxsrInterface.h
./passes/mxsr2msr/mxsr2msrSkeletonBuilderInterface.h
./passes/mxsr2msr/mxsr2msrTranslatorInterface.cpp
./passes/mxsr2msr/mxsr2msrTranslatorInterface.h
./passes/mxsr2msr/mxsr2msrSkeletonBuilderInterface.cpp
./passes/msr2msr/msr2msrInterface.h
./passes/msr2msr/msr2msrInterface.cpp
./passes/lpsr2lilypond/lpsr2lilypondInterface.h
./passes/lpsr2lilypond/lpsr2lilypondInterface.cpp
./passes/msr2lpsr/msr2lpsrInterface.cpp
./passes/msr2lpsr/msr2lpsrInterface.h
./passes/bsr2braille/bsr2brailleTranslatorInterface.h
./passes/bsr2braille/bsr2brailleTranslatorInterface.cpp
./passes/msr2bsr/msr2bsrInterface.h
./passes/msr2bsr/msr2bsrInterface.cpp
./passes/musicxml2mxsr/musicxml2mxsrInterface.h
./passes/musicxml2mxsr/musicxml2mxsrInterface.cpp
./passes/mxsr2guido/mxsr2guidoTranlatorInterface.h
./passes/mxsr2guido/mxsr2guidoTranlatorInterface.cpp
./converters/msr2guido/msr2guidoInterface.h
./converters/msr2guido/msr2guidoInterface.cpp
./converters/msr2braille/msr2brailleInterface.h
./converters/msr2braille/msr2brailleInterface.cpp
./converters/msdl2braille/msdl2brailleInterface.h
./converters/msdl2braille/msdl2brailleInterface.cpp
./converters/msdl2guido/msdl2guidoInterface.cpp
./converters/msdl2guido/msdl2guidoInterface.h
./converters/msdl2musicxml/msdl2musicxmlInterface.h
./converters/msdl2musicxml/msdl2musicxmlInterface.cpp
./converters/msdl2lilypond/msdl2lilypondInterface.h
./converters/msdl2lilypond/msdl2lilypondInterface.cpp
./converters/musicxml2braille/musicxml2brailleInterface.cpp
./converters/musicxml2braille/musicxml2brailleInterface.h
./converters/msr2lilypond/msr2lilypondInterface.cpp
./converters/msr2lilypond/msr2lilypondInterface.h
./converters/msr2musicxml/msr2musicxmlInterface.cpp
./converters/msr2musicxml/msr2musicxmlInterface.h
./converters/musicxml2musicxml/musicxml2musicxmlInterface.h
./converters/musicxml2musicxml/musicxml2musicxmlInterface.cpp
./converters/musicxml2lilypond/musicxml2lilypondInterface.h
./converters/musicxml2lilypond/musicxml2lilypondInterface.cpp
./converters/musicxml2guido/musicxml2guidoInterface.cpp
./converters/musicxml2guido/musicxml2guidoInterface.h
```

The converters are implemented as functions as well as CLI services that use the latter.

MusicFormats includes support for components versions numbering and history, see chapter **??** [MusicFormats components], page **??**.

`src/components/mfcComponents.h` includes all the components's header files.

5.2 Formats

The formats are in src/formats:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formats/
total 32
 0 drwxr-xr-x   10 jacquesmenu  staff    320 Jun 25 05:39:49 2021 ./
 0 drwxr-xr-x   13 jacquesmenu  staff    416 Jun 17 17:16:37 2021 ../
24 -rw-r--r--    1 jacquesmenu  staff  10244 Jun 19 07:58:55 2021 .DS_Store
 0 drwxr-xr-x   60 jacquesmenu  staff   1920 Jun 18 07:32:14 2021 bsr/
 0 drwxr-xr-x   42 jacquesmenu  staff   1344 May 26 08:20:55 2021 lpsr/
 0 drwxr-xr-x   12 jacquesmenu  staff    384 Apr 22 15:49:23 2021 msdl/
 0 drwxr-xr-x   10 jacquesmenu  staff    320 May 26 08:20:55 2021 msdr/
 0 drwxr-xr-x  151 jacquesmenu  staff   4832 Jun 20 09:58:00 2021 msr/
 0 drwxr-xr-x    6 jacquesmenu  staff    192 May 26 08:20:55 2021 mxsr/
```

The formats interfaces are in files with the format's name:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formats/bsr/bsr.*
8 -rw-r--r--@ 1 jacquesmenu  staff   700 Jun  6 06:35:19 2021 formats/bsr/bsr.cpp
8 -rw-r--r--@ 1 jacquesmenu  staff  1206 Jun 18 10:04:45 2021 formats/bsr/bsr.h

jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formats/lpsr/lpsr.*
8 -rw-r--r--@ 1 jacquesmenu  staff   703 Jun  6 06:35:19 2021 formats/lpsr/lpsr.cpp
8 -rw-r--r--@ 1 jacquesmenu  staff  1004 Jun  6 06:35:19 2021 formats/lpsr/lpsr.h

jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formats/msdl/msdl.*
8 -rw-r--r--@ 1 jacquesmenu  staff  736 Jun  6 06:35:19 2021 formats/msdl/msdl.cpp
8 -rw-r--r--@ 1 jacquesmenu  staff  643 Jun  6 06:35:19 2021 formats/msdl/msdl.h

jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formats/msdr/msdr.*
8 -rw-r--r--@ 1 jacquesmenu  staff  709 Jun  6 06:35:19 2021 formats/msdr/msdr.cpp
8 -rw-r--r--@ 1 jacquesmenu  staff  531 Jun  6 06:35:19 2021 formats/msdr/msdr.h

jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formats/msr/msr.*
8 -rw-r--r--@ 1 jacquesmenu  staff   700 Jun  6 06:35:19 2021 formats/msr/msr.cpp
8 -rw-r--r--@ 1 jacquesmenu  staff  2410 Jun 20 09:58:38 2021 formats/msr/msr.h

jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formats/mxsr/mxsr.*
8 -rw-r--r--@ 1 jacquesmenu  staff  3292 Jun  6 06:35:19 2021 formats/mxsr/mxsr.cpp
8 -rw-r--r--@ 1 jacquesmenu  staff  1555 Jun  6 06:35:19 2021 formats/mxsr/mxsrGeneration.
    h
```

5.3 Representations

The representations are in src/representations:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll representations/
total 24
 0 drwxr-xr-x   11 jacquesmenu  staff    352 Dec 30 17:25:10 2021 ./
 0 drwxr-xr-x   18 jacquesmenu  staff    576 Jan 16 16:50:25 2022 ../
24 -rw-r--r--@   1 jacquesmenu  staff  10244 Jan  6 17:40:44 2022 .DS_Store
 0 drwxr-xr-x    8 jacquesmenu  staff    256 Dec 30 10:26:26 2021 braille/
 0 drwxr-xr-x   69 jacquesmenu  staff   2208 Jan  4 07:52:14 2022 bsr/
 0 drwxr-xr-x    4 jacquesmenu  staff    128 Dec 30 10:27:01 2021 guido/
 0 drwxr-xr-x   51 jacquesmenu  staff   1632 Jan  4 07:52:36 2022 lpsr/
 0 drwxr-xr-x   16 jacquesmenu  staff    512 Jan  4 07:52:55 2022 msdl/
 0 drwxr-xr-x   12 jacquesmenu  staff    384 Jan  4 07:53:13 2022 msdr/
 0 drwxr-xr-x  165 jacquesmenu  staff   5280 Jan  4 07:53:34 2022 msr/
 0 drwxr-xr-x   10 jacquesmenu  staff    320 Jan  4 07:53:54 2022 mxsr/
```

5.4 Passes

The passs are in src/passes:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll passes
total 24
 0 drwxr-xr-x  14 jacquesmenu  staff    448 Nov 24 16:29:20 2021 ./
 0 drwxr-xr-x  20 jacquesmenu  staff    640 Nov 16 08:12:03 2021 ../
24 -rw-r--r--@  1 jacquesmenu  staff  10244 Nov 24 10:38:11 2021 .DS_Store
 0 drwxr-xr-x   8 jacquesmenu  staff    256 Oct 22 07:19:11 2021 bsr2braille/
 0 drwxr-xr-x   6 jacquesmenu  staff    192 Oct 22 07:20:34 2021 bsr2bsr/
 0 drwxr-xr-x  10 jacquesmenu  staff    320 Nov 16 10:09:27 2021 lpsr2lilypond/
 0 drwxr-xr-x  14 jacquesmenu  staff    448 Oct 22 07:22:09 2021 msdl2msr/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 Oct 22 07:24:35 2021 msr2bsr/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 Nov  1 16:31:34 2021 msr2lpsr/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 Nov  1 16:31:34 2021 msr2msr/
 0 drwxr-xr-x   6 jacquesmenu  staff    192 Oct 22 07:27:46 2021 msr2mxsr/
 0 drwxr-xr-x   4 jacquesmenu  staff    128 Oct 22 07:28:37 2021 mxsr2guido/
 0 drwxr-xr-x  10 jacquesmenu  staff    320 Nov  1 16:31:34 2021 mxsr2msr/
 0 drwxr-xr-x   4 jacquesmenu  staff    128 Oct 22 07:29:50 2021 mxsr2musicxml/
```

Some passes are named translators (converters could have been used), and others are not. In src/passes/mxsr2msr/, class `mxsr2msrSkeletonBuilder` does not translate MusicXML data to another full representation: it merely creates a skeleton containing voices, are are empty:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll passes/mxsr2msr/
total 1808
   0 drwxr-xr-x   8 jacquesmenu  staff     256 Jun 25 05:47:41 2021 ./
   0 drwxr-xr-x  16 jacquesmenu  staff     512 May 26 08:20:55 2021 ../
  96 -rw-r--r--@  1 jacquesmenu  staff   48389 Jun 21 07:43:20 2021 mxsr2msrOah.cpp
  40 -rw-r--r--@  1 jacquesmenu  staff   20327 Jun 16 10:41:37 2021 mxsr2msrOah.h
 192 -rw-r--r--@  1 jacquesmenu  staff   97896 Jun 25 08:58:38 2021
    mxsr2msrSkeletonBuilder.cpp
  48 -rw-r--r--@  1 jacquesmenu  staff   20942 Jun 25 07:36:29 2021
    mxsr2msrSkeletonBuilder.h
1280 -rw-r--r--@  1 jacquesmenu  staff  651474 Jun 25 07:49:52 2021 mxsr2msrTranslator.cpp
 152 -rw-r--r--@  1 jacquesmenu  staff   77039 Jun 21 07:43:20 2021 mxsr2msrTranslator.h
```

The passes functionality is available as functions in `*Interface.*`:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > look Interface
./representations/msr/msrInterface.cpp
./representations/msr/msrInterface.h
./representations/lpsr/lpsrInterface.cpp
./representations/lpsr/lpsrInterface.h
./representations/bsr/bsrInterface.h
./representations/bsr/bsrInterface.cpp
./passes/mxsr2musicxml/mxsr2musicxmlTranlatorInterface.h
./passes/mxsr2musicxml/mxsr2musicxmlTranlatorInterface.cpp
./passes/bsr2bsr/bsr2bsrFinalizerInterface.h
./passes/bsr2bsr/bsr2bsrFinalizerInterface.cpp
./passes/msr2mxsr/msr2mxsrInterface.cpp
./passes/msr2mxsr/msr2mxsrInterface.h
./passes/mxsr2msr/mxsr2msrSkeletonBuilderInterface.h
./passes/mxsr2msr/mxsr2msrTranslatorInterface.cpp
./passes/mxsr2msr/mxsr2msrTranslatorInterface.h
./passes/mxsr2msr/mxsr2msrSkeletonBuilderInterface.cpp
./passes/msr2msr/msr2msrInterface.h
./passes/msr2msr/msr2msrInterface.cpp
./passes/lpsr2lilypond/lpsr2lilypondInterface.h
./passes/lpsr2lilypond/lpsr2lilypondInterface.cpp
./passes/msr2lpsr/msr2lpsrInterface.cpp
./passes/msr2lpsr/msr2lpsrInterface.h
./passes/bsr2braille/bsr2brailleTranslatorInterface.h
./passes/bsr2braille/bsr2brailleTranslatorInterface.cpp
```

```
./passes/msr2bsr/msr2bsrInterface.h
./passes/msr2bsr/msr2bsrInterface.cpp
./passes/musicxml2mxsr/musicxml2mxsrInterface.h
./passes/musicxml2mxsr/musicxml2mxsrInterface.cpp
./passes/mxsr2guido/mxsr2guidoTranlatorInterface.h
./passes/mxsr2guido/mxsr2guidoTranlatorInterface.cpp
./converters/msr2guido/msr2guidoInterface.h
./converters/msr2guido/msr2guidoInterface.cpp
./converters/msr2braille/msr2brailleInterface.h
./converters/msr2braille/msr2brailleInterface.cpp
./converters/msdl2braille/msdl2brailleInterface.h
./converters/msdl2braille/msdl2brailleInterface.cpp
./converters/msdl2guido/msdl2guidoInterface.cpp
./converters/msdl2guido/msdl2guidoInterface.h
./converters/msdl2musicxml/msdl2musicxmlInterface.h
./converters/msdl2musicxml/msdl2musicxmlInterface.cpp
./converters/msdl2lilypond/msdl2lilypondInterface.h
./converters/msdl2lilypond/msdl2lilypondInterface.cpp
./converters/musicxml2braille/musicxml2brailleInterface.cpp
./converters/musicxml2braille/musicxml2brailleInterface.h
./converters/msr2lilypond/msr2lilypondInterface.cpp
./converters/msr2lilypond/msr2lilypondInterface.h
./converters/msr2musicxml/msr2musicxmlInterface.cpp
./converters/msr2musicxml/msr2musicxmlInterface.h
./converters/musicxml2musicxml/musicxml2musicxmlInterface.h
./converters/musicxml2musicxml/musicxml2musicxmlInterface.cpp
./converters/musicxml2lilypond/musicxml2lilypondInterface.h
./converters/musicxml2lilypond/musicxml2lilypondInterface.cpp
./converters/musicxml2guido/musicxml2guidoInterface.cpp
./converters/musicxml2guido/musicxml2guidoInterface.h
```

5.5 Generators

A generator is a multi-pass command line service that creates an ouput from scratch, without reading anything. All of them use `src/formatsgeneration/multiGeneration/multiGeneration.h/.cpp` to offer a set of output formats:

- `src/clisamples/Mikrokosmos3Wandering.cpp`
  creates a score for this Bartok piece in various forms, depending on the options. It has been used to check th e MSR API's:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formatsgeneration/
    multiGeneration/
total 56
 0 drwxr-xr-x   4 jacquesmenu  staff    128 Apr 22 15:49:16 2021 ./
 0 drwxr-xr-x  10 jacquesmenu  staff    320 May 26 08:20:55 2021 ../
40 -rw-r--r--@  1 jacquesmenu  staff  16774 Jun  6 06:38:55 2021 multiGenerationOah.
    cpp
16 -rw-r--r--@  1 jacquesmenu  staff   6750 Jun  6 06:38:55 2021 mfMultiGenerationOah
    .h
```

  For example:

```
jacquesmenu@macmini: ~ > Mikrokosmos3Wandering -lilypond -a
What LilyPondIssue34 does:

    This multi-pass generator creates a textual representation
    of the LilyPondIssue34 score.
    It basically performs 4 passes when generating LilyPond output output:

        Pass 1:  generate a first MSR for the LilyPondIssue34 score
        Pass 2:  converts the first MSR a second MSR;
        Pass 3:  converts the second MSR into a
                 LilyPond Score Representation (LPSR);
        Pass 4:  converts the LPSR to LilyPond code
                 and writes it to standard output.

    Other passes are performed according to the options, such as
    displaying views of the internal data or printing a summary of the score.

    The activity log and warning/error messages go to standard error.
```

- src/clisamples/LilyPondIssue34.cpp creates a score for the LilyPond issue #34 issue, also in various forms:;

```
jacquesmenu@macmini: ~ > LilyPondIssue34 -musicxml -a
What LilyPondIssue34 does:

    This multi-pass generator creates a textual representation
    of the LilyPondIssue34 score.
    It basically performs 4 passes when generating MusicXML output output:

        Pass 1:  generate a first MSR for the LilyPondIssue34 score
        Pass 2:  converts the first MSR a second MSR, to apply options;
        Pass 3:  converts the second MSR into an MusicXML tree;
        Pass 4:  converts the MusicXML tree to MusicXML code
                 and writes it to standard output.

    Other passes are performed according to the options, such as
    displaying views of the internal data or printing a summary of the score.

    The activity log and warning/error messages go to standard error.
```

5.6 Converters

The MusicFormats converters chain passes into a sequence, each pass reading the input or the format produced by the preceeding one:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll converters/
total 32
 0 drwxr-xr-x  17 jacquesmenu  staff    544 May 26 08:20:55 2021 ./
 0 drwxr-xr-x  13 jacquesmenu  staff    416 Jun 17 17:16:37 2021 ../
24 -rw-r--r--   1 jacquesmenu  staff  10244 Jun 18 10:34:45 2021 .DS_Store
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msdl2braille/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msdl2guido/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msdl2lilypond/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msdl2musicxml/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msdlconverter/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msr2braille/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msr2guido/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msr2lilypond/
 0 drwxr-xr-x   8 jacquesmenu  staff    256 May 26 08:20:55 2021 msr2musicxml/
 0 drwxr-xr-x   4 jacquesmenu  staff    128 May 26 08:20:55 2021 musicxml2braille/
 0 drwxr-xr-x   4 jacquesmenu  staff    128 May 26 08:20:55 2021 musicxml2guido/
 0 drwxr-xr-x   4 jacquesmenu  staff    128 May 26 08:20:55 2021 musicxml2lilypond/
 0 drwxr-xr-x   4 jacquesmenu  staff    128 May 26 08:20:55 2021 musicxml2musicxml/
```

5.7 Running a service

When a MusicFormats service is *run* from the command line or through an API function, an instance of class `mfServiceRunData` is created.

This class is defined in src/mflibrary/mfServiceRunData.h/.cpp to hold data specific to the run. They are global data, but don't belong to the regular, invariant data contained in the library, such as the notes pitches in various languages:

```
class EXP mfServiceRunData : public smartable
{
  public:

    // creation
    // ------------------------------------------------------

    static SMARTP<mfServiceRunData> create (const std::string& serviceName);

    static SMARTP<mfServiceRunData> create (
                            const std::string& serviceName,
                            int            argc,
                            char*          argv[]);

    static SMARTP<mfServiceRunData> create (
                            const std::string&      serviceName,
                            mfOptionsAndArguments& optionsAndArguments);

  public:

    // constructors/destructor
    // ------------------------------------------------------

                          mfServiceRunData (const std::string& serviceName);

                          mfServiceRunData (
                            const std::string& serviceName,
                            int            argc,
                            char*          argv[]);

                          mfServiceRunData (
                            const std::string&      serviceName,
                            mfOptionsAndArguments& optionsAndArguments);

    virtual               ~mfServiceRunData ();

  // .. .. ..

  private:

    // private fields
    // ------------------------------------------------------

    // service name
    std::string           fServiceName;

    // conversion date
    std::string           fRunDateFull;
    std::string           fRunDateYYYYMMDD;

    // conversion command
    std::string           fCommandAsSupplied;

    std::string           fCommandWithLongOptionsNames;
    std::string           fCommandWithShortOptionsNames;

    // options and arguments
    mfOptionsAndArguments
                          fOptionsAndArguments;

    // command line
    std::string           fCommandLineAsSupplied;

    // input source
    std::string           fInputSourceName;
};
```

The various constructors are used depending on the way the service is run.

For example, if is created this way in src/clisamples/xml2ly.cpp:

```
int main (int argc, char* argv[])
{
  // setup signals catching
  // ------------------------------------------------------

// JMI  catchSignals ();

  // the service name
  // ------------------------------------------------------

  std::string serviceName = argv [0];

  // create the global output and log indented streams
  // ------------------------------------------------------

  createTheGlobalIndentedOstreams (std::cout, std::cerr);

  // create the global run data
  // ------------------------------------------------------

  gServiceRunData =
    mfServiceRunData::create (serviceName);

  // ... ... ...
}
```

Then the various run data can be accessed easily:

```
  std::string
    inputSourceName =
      gServiceRunData->getInputSourceName ();
```

The run date is used for example in class `lpsrScore`, defined in src/formats/lpsr//lpsrScores.h/.cpp:

```
lpsrScore::lpsrScore (
  int                    inputLineNumber,
  const S_msrScore&      theMsrScore,
  const S_mfcMultiComponent& multiComponent)
    : lpsrElement (inputLineNumber)
{
  // ...

  fMultiComponent = multiComponent;

  // should the initial comments about the service and the options used
  // be generated?
  if (gGlobalLpsr2lilypondOahGroup->getXml2lyInfos ()) {
    // create the 'generated by' comment
    {
      std::stringstream ss;

      ss <<
        "Generated by " <<
        gOahOahGroup->getOahOahGroupServiceName () <<
        ' ' <<
        getGlobalMusicFormatsVersionNumberAndDate () <<
        std::endl <<

        "% on " <<
        gServiceRunData->getRunDateFull () <<
        std::endl <<

        "% from ";

      std::string inputSourceName =
        gServiceRunData->getInputSourceName ();

      if (inputSourceName == "-") {
        ss << "standard input";
      }
      else {
        ss << "\"" << inputSourceName << "\"";
      }

      fInputSourceNameComment =
        lpsrComment::create (
          inputLineNumber,
          ss.str (),
          lpsrCommentGapAfterwardsKind::kCommentGapAfterwardsYes);
    }
  }

  // ...
}
```

Chapter 6

Command line samples

The src/clisamples folder contains example of the use of MusicFormats in CLI services. They are out of the library proper, and built with a specific `Makefile`:

```
jacquesmenu@macmini: ~/musicformats-git-dev > ll samples/
total 320
 0 drwxr-xr-x  16 jacquesmenu  staff    512 Jun 29 09:59:07 2021 ./
 0 drwxr-xr-x  28 jacquesmenu  staff    896 Jul  1 05:37:35 2021 ../
16 -rw-r--r--   1 jacquesmenu  staff   6148 May 26 08:20:55 2021 .DS_Store
 8 -rw-r--r--   1 jacquesmenu  staff    116 Apr 22 15:49:06 2021 .gitignore
40 -rw-r--r--@  1 jacquesmenu  staff  18344 Jun 29 11:05:18 2021 LilyPondIssue34.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff   2101 Jun 29 10:00:56 2021 Makefile
40 -rw-r--r--@  1 jacquesmenu  staff  18362 Jun 29 11:05:10 2021 Mikrokosmos3Wandering.cpp
24 -rw-r--r--@  1 jacquesmenu  staff  10017 May 31 11:12:12 2021 MusicAndHarmonies.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff   3117 May 31 11:17:27 2021 libMultipleInitsTest.cpp
48 -rw-r--r--@  1 jacquesmenu  staff  21459 Jun 29 11:05:02 2021 msdlconverter.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff    898 May 31 11:15:59 2021 musicformatsversion.cpp
24 -rw-r--r--@  1 jacquesmenu  staff   8642 Jun 28 07:42:57 2021 xml2Any.cpp
24 -rw-r--r--@  1 jacquesmenu  staff  10085 Jul  1 06:22:13 2021 xml2brl.cpp
24 -rw-r--r--@  1 jacquesmenu  staff  10519 Jul  1 06:22:50 2021 xml2gmn.cpp
24 -rw-r--r--@  1 jacquesmenu  staff  10320 Jul  1 00:09:51 2021 xml2ly.cpp
24 -rw-r--r--@  1 jacquesmenu  staff  10473 Jul  1 06:21:10 2021 xml2xml.cpp
```

All the `*.cpp` files contain a `main ()` function using the interfaces for their purpose. Among them:

- `libMultipleInitsTest.cpp` is a maintainance tool to check that the MusicFormats library is not initialized more that once;
- `MusicAndHarmonies.cpp` creates a score at random with harmonies in it;
- `Mikrokosmos3Wandering.cpp` and `LilyPondIssue34.cpp` are generators;
- `xml2Any.cpp` uses the `oahOptionsVector` way to supply arguments instead of `arg/argv`;
- `xml2ly`, `xml2brl`, `xml2xml` and `xml2gmn` are converters from MusicXML to other formats;
- `msdlconverter.cpp` it the MSDL converter.

Chapter 7

Warning and errors (WAE)

Warning and errors in MusicFormats are handled with a set of functions defined in the `wae` folder.

class `mfException` and context-specific exceptions are defined in `src/wae/waeExceptions`, such as:

```
//______________________________________________________________________________
class EXP mfOahException: public mfException
{
  public:
    mfOahException (std::string const& exceptionDescription) throw ()
      : mfException (exceptionDescription)
    {}
};
typedef SMARTP<mfOahException> S_mfOahException;
```

A typical use of exceptions in `src/passes/lpsr2lilypond/lpsr2lilypondInterface.cpp` is:

```
    // convert the LPSR into LilyPond code
    try {
      translateLpsrToLilypond (
        theLpsrScore,
        gMsrOahGroup,
        gLpsrOahGroup,
        passIDKind,
        passDescription,
        lilypondStandardOutputStream);
    } // try

    catch (lpsr2lilypondException& e) {
      mfDisplayException (e, gOutput);
      return;
    }
    catch (std::exception& e) {
      mfDisplayException (e, gOutput);
      return;
    }
```

One finds in `src/wae/mfEnableAbortToDebugErrorsSetting.h` the `MF_ABORT_TO_DEBUG_ERRORS_IS_ENABLED` macro to help debugging the code base:

```
// comment the following definition if abort on internal errors is desired
// CAUTION: DON'T USE THIS IN PRODUCTION CODE,
// since that could kill a session on a \Web\ server, for example

#ifndef MF_ABORT_TO_DEBUG_ERRORS_IS_ENABLED
  #define MF_ABORT_TO_DEBUG_ERRORS_IS_ENABLED
#endif // MF_ABORT_TO_DEBUG_ERRORS_IS_ENABLED
```

Chapter 8

The trace facility

MusicFormats is instrumented with an optionnal, full-fledged trace facility, with numerous options to display what is going on when using the library. One can build the library with or without trace, which applies to the whole code base.

8.1 Activating the trace

Trace is controlled by MF_TRACE_IS_ENABLED, defined or nor in src/oah/mfPreprocessorSettings.h:

```
#ifndef ___enableTraceIfDesired___
#define ___enableTraceIfDesired___

#ifndef MF_TRACE_IS_ENABLED
  // comment the following definition if no trace is desired
  #define MF_TRACE_IS_ENABLED
#endif // MF_TRACE_IS_ENABLED

#endif // ___enableTraceIfDesired___
```

This file should be included when the trace facility is used:

```
#include "mfPreprocessorSettings.h"
```

The files src/oah/TraceOah.h/.cpp contain the options to the trace facility itself.

Be sure to build MusicFormats with MF_TRACE_IS_ENABLED both active and commented out before creating a new v* version branch, to check that variables scopes are fine.

For example, xml2ly -insider -help-tracexml2lyoption -insider produces:

```
menu@macbookprojm > xml2ly -insider -help-trace
--- Help for group "OAH Trace" ---
  OAH Trace (-ht, -help-trace) (use this option to show this group)
    There are trace options transversal to the successive passes,
      showing what's going on in the various translation activities.
      They're provided as a help for the maintainance of MusicFormats,
      as well as for the curious.
      The options in this group can be quite verbose, use them with small input data!
      All of them imply '-trace-passes, -tpasses'.
  --------------------------
    Options handling trace    (-htoh, -help-trace-options-handling):
      -toah, -trace-oah
            Write a trace of options and help handling to standard error.
      -toahd, -trace-oah-details
```

```
            Write a trace of options and help handling with more details to standard error
    .
    Score to voices            (-htstv, -help-trace-score-to-voices):
      -t<SHORT_NAME>, -trace-<LONG_NAME>
            Trace SHORT_NAME/LONG_NAME in books to voices.
      The 10 known SHORT_NAMEs are:
        book, scores, pgroups, pgroupsd, parts, staves, st, schanges,
        .
      The 10 known LONG_NAMEs are:
        -books, -scores, -part-groups, -part-groups-details,
        -parts, -staves, -staff-details, -staff-changes, -voices and
        -voices-details.
... ... ...
```

8.2 Trace categories

8.3 Using traces in practise

In src/passes/lpsr2lilypond/lpsr2lilypondTranslator.cpp, the trace for the generation of LilyPond code for a regular note in a measure is produced by:

```
void lpsr2lilypondTranslator::generateCodeForNoteInMeasure (
  const S_msrNote& note)
{
  int inputLineNumber =
    note->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceNotes ()) {
    std::stringstream ss;

    ss <<
      std::endl <<
      "% --> generating code for noteRegularInMeasure " <<
      note->asString () <<
      ", line " << inputLineNumber <<
      std::endl;

    gLog          << ss.str ();
    fLilypondCodeStream << ss.str ();
  }
#endif // MF_TRACE_IS_ENABLED
```

8.4 Debugging traces handling

If case there is a null pointer in a case such as:

```
    gMsrOahGroup->getUseFilenameAsWorkCreditTypeTitle ()
```

the way to go is to:

- locate gGlobalMxsr2msrOahGroup in the *.h it is declared in;
- check that the creation method in the same, such as createGlobalMxsr2msrOahGroup (), is called in the *InsiderHandler.cpp file for the service that crashed, which may require including that .h header in InsiderHandler.cpp.

Chapter 9

Multi-lingual support

MusicFormats supports multiple languages in various areas:

- note pitches names in MSR, LPSR and generated LilyPond code;
- LilyPond chord names;
- reserved keywords in MSDL.

Chapter 10

Textual input and output

10.1 Indented output streams

To meet the need of indented output to produce , we got inspiration from https://stackoverflow.com/questions/2212776/overload-handling-of-stdendl.

This leads to class `mfIndentedOstream`, defined in src/utilities/mfIndentedTextOutput.h/.cpp:

```
class EXP mfIndentedOstream: public std::ostream, public smartable
```

The key to this lies in thw overloaded method `mfIndentedOStreamBuf::sync ()`:

```
//------------------------------------------------------------------------
int mfIndentedOStreamBuf::sync ()
{
  // When we sync the stream with fOutputSteam:
  // 1) output the indentation then the buffer
  // 2) reset the buffer
  // 3) flush the actual output stream we are using.

  unsigned int strSize = str ().size ();

  // fetch the last non-space character in the buffer
  // caution: the std::endl is present as the last character!
  size_t found = str ().find_last_not_of (' ', strSize - 2);

  // this can be uncommented to see low level informations
  // fOutputSteam << "% strSize: " << strSize << ", found: " << found << std::endl;

  // output the indenter
  fOutputSteam << fOutputIndenter;

  // output the buffer
  if (found == strSize - 3) {
    // don't output the trailing spaces, but output the end of line
    fOutputSteam << str ().substr (0, found + 1) << std::endl;
  }
  else {
    // output the whole buffer
    fOutputSteam << str ();
  }

  // reset the buffer
  str ("");

  // flush the output stream
  fOutputSteam.flush ();

  return 0;
}
```

Such indented output streams are used for nearly all of the output produced by MusicFormats, except for Braille which follows its own rules for indentation of cell lines.

10.2 Creating indented output streams

Such indented output streams are passed by reference to various methods which only know of `osstream`, among them:

```
    void                      print (std::ostream& os) const override;
```

All those methods manipulate `mfIndentedOstream` instances seamlessly, not knowing their actual type. This is needed for the use of MusicFormats by applications through the API and not as an service. In particular, Web sites also know only of regular output streams.

So when and where are `mfIndentedOstream` instances created?

Functions `translateLpsrToLilypondWithHandler ()`, in `src/passes/lpsr2lilypond/lpsr2lilypondInterface.` creates one depending on wether it writes the LilyPond code to standard output or to a file.

The parameters to this function are:

```
EXP void translateLpsrToLilypondWithHandler (
  const S_lpsrScore& theLpsrScore,
  const S_msrOahGroup&     msrOpts,
  const S_lpsrOahGroup&    lpsrOpts,
  mfPassIDKind           passIDKind,
  const std::string&     passDescription,
  S_oahHandler      handler,
  std::ostream&          out,
  std::ostream&          err)
```

In order to have a global current indentation, MusicFormats uses global variable `gIndenter`, because it should otherwise passed over to many methods throughout the code base. It is defined in `src/mfutilities/mfIndente`

```
#define gIndenter mfOutputIndenter::gGlobalOStreamIndenter
```

When writing to standard output, the indented output stream is constructed above the caller-supplied `out`:

```
    // create an indented output stream for the LilyPond code
    // to be written to outputFileStream
    mfIndentedOstream
      lilypondStandardOutputStream (
        out,
        gIndenter);

    // convert the LPSR into LilyPond code
    try {
      translateLpsrToLilypond (
        theLpsrScore,
        gMsrOahGroup,
        gLpsrOahGroup,
        passIDKind,
        passDescription,
        lilypondStandardOutputStream);
    }
```

When writing to a file, an **std::ofstream** is instantiated to write to the file given by its name, and the indented output stream is constructed above the latter:

```
     std::ofstream
      outputFileStream (
        outputFileName.c_str (),
        std::ofstream::out);

    // create an indented output stream for the LilyPond code
    // to be written to outputFileStream
    mfIndentedOstream
      lilypondFileOutputStream (
        outputFileStream,
        gIndenter);

    // convert the LPSR into LilyPond code
    try {
      translateLpsrToLilypond (
        theLpsrScore,
        gMsrOahGroup,
        gLpsrOahGroup,
        passIDKind,
        passDescription,
        lilypondFileOutputStream);
    }
```

The code that uses MusicFormats thus does not have to care for indented streams instantiation: this is done behind the scene by the library.

10.3 Indenting the output

Indenting the output is handled with a single variable defined in src/mfutilities/mfIndentedTextOutput.h. This sharing of a global variable is needed to produce orderly output, since many parts of the MusicFormats library can contribute to it:

```
// useful shortcut macros
#define gIndenter mfOutputIndenter::gGlobalOStreamIndenter
#define gTab      mfOutputIndenter::gGlobalOStreamIndenter.getSpacer ()
```

A typical sequence to produce indented output is:

```
void msrTransposition::print (std::ostream& os) const
{
  const int fieldWidth = 22;

  os <<
    "Transposition" <<
    ", line " << fInputStartLineNumber <<
    std::endl;

  ++gIndenter;

  os << std::left <<
    std::setw (fieldWidth) <<
    "fTranspositionDiatonic" << ": " << fTranspositionDiatonic <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionChromatic" << ": " << fTranspositionChromatic <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionOctaveChange" << ": " << fTranspositionOctaveChange <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionDouble" << ": " << fTranspositionDouble <<
    std::endl << std::endl;

  --gIndenter;
}
```

Note that the new value of `gIndenter` after `++gIndenter` and `--gIndenter` is taken into account only at the *next* end-of-line – the output of lines 5 to 8 above is indented one level less that the various field values output afterwards.

There can be indentation issues, in which the user gets messages like:

```
% ### Indentation has become negative: -1
```

To debug this:

- activate the debugging for the indenter, uncommenting this definition in `src/components/mfIndentedTextOutp`

  ```
  //______________________________________________________________________________
  //#define DEBUG_INDENTER
  ```

- activate abortion on errors in `src/wae/mfEnableAbortToDebugErrorsSetting.h`:

  ```
  // comment the following definition if abort on internal errors is desired
  // CAUTION: DON'T USE THIS IN PRODUCTION CODE,
  // since that could kill a session on a web server, for example

  #ifndef MF_ABORT_TO_DEBUG_ERRORS_IS_ENABLED
    //#define MF_ABORT_TO_DEBUG_ERRORS_IS_ENABLED
  #endif MF_ABORT_TO_DEBUG_ERRORS_IS_ENABLED
  ```

10.4 Printing descriptions

There is a standard set of methods to print the contents of the descriptions in MusicFormats to standard output, depending on the granularity of the information to be displayed:

```
    void                      print (std::ostream& os) const override;

    std::string               asString () const override;
    std::string               asStringShort () const override;
```

There are also more specific methods such as:

```
    void                      printFull (std::ostream& os) const override;

    void                      printSummary (std::ostream& os) const override;
```

Note that:

- virtual method `asString ()` produces a rather condensed view of the data to be displayed as part of a single line;
- virtual method `print ()` may produce its output on multiples lines, which always ends with an end of line.

Most classes in MusicFormats can be printed with the `<<` operator. Since MusicFormats is a large, a test is done for nullity, which is safer and easier for the applications that use it:

```
std::ostream& operator << (std::ostream& os, const S_msrElement& elt)
{
  if (elt) {
    elt->print (os);
  }
  else {
    os << "[NULL]" << std::endl;
  }

  return os;
}
```

In simple cases, virtual method `print ()` merely calls virtual method `asString ()`:

```
void msrElement::print (std::ostream& os) const
{
  os << asString () << std::endl;
}
```

All `asString ()` methods produce an output of the form `[...]`, in order to facilitate selecting the whole with a double click to help the user, since such output can be nested:

```
std::string msrTransposition::asString () const
{
  std::stringstream ss;

  ss <<
    "[Transposition" <<
    ", fTranspositionDiatonic = " << fTranspositionDiatonic <<
    ", fTranspositionChromatic = " << fTranspositionChromatic <<
    ", fTranspositionOctaveChange = " << fTranspositionOctaveChange <<
    ", fTranspositionDouble = " << fTranspositionDouble <<
    ", line " << fInputStartLineNumber <<
    ']';

  return ss.str ();
}
```

A typical sequence to produce indented output is:

```
void msrTransposition::print (std::ostream& os) const
{
  const int fieldWidth = 22;

  os <<
    "Transposition" <<
    ", line " << fInputStartLineNumber <<
    std::endl;

  ++gIndenter;

  os << std::left <<
    std::setw (fieldWidth) <<
    "fTranspositionDiatonic" << ": " << fTranspositionDiatonic <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionChromatic" << ": " << fTranspositionChromatic <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionOctaveChange" << ": " << fTranspositionOctaveChange <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionDouble" << ": " << fTranspositionDouble <<
    std::endl << std::endl;

  --gIndenter;
}
```

The global indenter is `gIndenter`:

```
// the global output indenter for general use
EXP mfOutputIndenter& getGlobalOutputIndenter ();

// useful shortcut macros
#define gIndenter getGlobalOutputIndenter ()
#define gTab      getGlobalOutputIndenter ().getSpacer ()
```

The main indented output streams are `gOutput` and `gLog`:

```
// the global log and output indented streams
EXP extern S_mfIndentedOstream gGlobalOutputIndentedOstream;
EXP extern S_mfIndentedOstream gGlobalLogIndentedOstream;

#define gOutput *gGlobalOutputIndentedOstream
#define gLog    *gGlobalLogIndentedOstream
```

Chapter 11

Binary data output

Binary data output is done for Braille

Chapter 12

CPU measurements

Option `-cpu` displays the time spent in the successive passes, such as:

```
Activity  Description                                                 Kind       CPU (sec)
--------  ----------------------------------------------------------  ---------  ---------

          Handle the options and arguments from argc/argv             mandatory    0.01187
Pass 1    Create an MXSR from a MusicXML file                  mandatory    0.00471
Pass 2a   Create an MSR skeleton from the MXSR                    mandatory    0.00222
Pass 2b   Populate the MSR skeleton from MusicXML data                mandatory    0.00405
Pass 4    Convert the MSR into an LPSR                                mandatory    0.00137
Pass 5    Convert the LPSR into LilyPond code                     mandatory    0.00136

Total (sec)  Mandatory  Optional
-----------  ---------  ---------
0.02558      0.02558    0.00000
```

These numbers are for the CPU only, not including input and output tasks. The time spent in options handling is roughly always the same on a given machine.

Class `mfTimingItemsList`, defined in `src/utilities/mfTiming.h/.cpp`, provides:

```
class EXP mfTimingItemsList {
  // ... ... ...

  public:

    // global variable for general use
    // ------------------------------------------------------

    static mfTimingItemsList          sGlobalTimingItemsList;

  public:

    // public services
    // ------------------------------------------------------

    // add an item
    void                      appendTimingItem (
                                std::string           activity,
                                std::string           description,
                                mfTimingItemKind kind,
                                clock_t          startClock,
                                clock_t          endClock);

  // ... ... ...
}
```

Functions translateMsrToLpsrScore () in src/passes/msr2lpsr/msr2lpsrInterface.cpp measures time to perform the conversion this way:

```
S_lpsrScore translateMsrToLpsr (
  const S_msrScore&          originalMsrScore,
  const S_msrOahGroup&       msrOpts,
  const S_lpsrOahGroup&      lpsrOpts,
  mfPassIDKind               passIDKind,
  std::string                passDescription,
  const S_mfcMultiComponent& multiComponent)
{
  if (gGlobalLpsr2lilypondOahGroup->getNoLilypondCode ()) {
    gLog <<
      "Option '-nolpc, -no-lilypond-code' is set, no LPSR is created" <<
      std::endl;

    return nullptr;
  }

#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  mfAssert (
    __FILE__, __LINE__,
    originalMsrScore != nullptr,
    "originalMsrScore is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  // start the clock
  clock_t startClock = clock ();

  // set the global current passID
  setGlobalCurrentPassIDKind (passIDKind);

#ifdef MF_TRACE_IS_ENABLED
  if (gEarlyOptions.getEarlyTracePasses ()) {
    std::string separator =
      "%--------------------------------------------------------------";

    std::stringstream ss;

    ss <<
      std::endl <<
      separator <<
      std::endl <<
      gTab <<
      gLanguage->passIDKindAsString (passIDKind) << ": " << passDescription <<
      std::endl <<
      separator;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create an msr2lpsrTranslator
  msr2lpsrTranslator
    translator (
      originalMsrScore);

  // build the LPSR score
  S_lpsrScore
    resultingLpsr =
      translator.translateMsrToLpsr (
        originalMsrScore,
        multiComponent);

  clock_t endClock = clock ();

  // register time spent
  gGlobalTimingItemsList.appendTimingItem (
    passIDKind,
    passDescription,
    mfTimingItemKind::kMandatory,
    startClock,
    endClock);
```

Part II

Programming style and conventions

Chapter 13

Programming style and conventions

13.1 Files naming conventions

Most file names start with an identification of the component they belong to, such as 'oah', 'mxsr', 'msr', 'lpsr', 'lilypond', 'bsr', 'braille', 'xml2ly', 'xml2brl' and msdl.
The ancillary files such as src/utilities/mfIndentedTextOutput.h/.cpp follow this rule too, with an mf prefix.

The '*Oah.*' files handle the options and help for the corresponding component, such as 'src/passes/msr2msr/msr2msrOah.h/.cpp'.

The 'src/oah/traceOah.h/.cpp', src/oah/musicxmlOah.h/.cpp 'extra' and 'general' prefixes are about the corresponding help groups.

There are a couple of 'globlal' files not related to any particular component, placed in src/mfutilities/ with an mf name prefix:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll mfutilities/
total 200
 0 drwxr-xr-x  15 jacquesmenu  staff    480 Oct 22 06:25:57 2021 ./
 0 drwxr-xr-x  19 jacquesmenu  staff    608 Oct 22 05:29:29 2021 ../
 8 -rw-r--r--@  1 jacquesmenu  staff   3255 Oct 18 20:22:16 2021 mfBool.cpp
16 -rw-r--r--@  1 jacquesmenu  staff   4917 Oct 18 19:56:51 2021 mfBool.h
 8 -rw-r--r--@  1 jacquesmenu  staff   1336 Oct 15 18:48:10 2021 mfEnumAll.h
16 -rw-r--r--@  1 jacquesmenu  staff   7182 Nov  8 13:08:51 2021 mfIndentedTextOutput.cpp
16 -rw-r--r--@  1 jacquesmenu  staff   7715 Nov  8 13:08:40 2021 mfIndentedTextOutput.h
 8 -rw-r--r--@  1 jacquesmenu  staff    889 Oct 15 20:34:47 2021 mfMusicformatsErrorKind.
   cpp
 8 -rw-r--r--@  1 jacquesmenu  staff    629 Oct 15 20:34:47 2021 mfMusicformatsErrors.h
 8 -rw-r--r--@  1 jacquesmenu  staff   2541 Nov  5 11:29:25 2021 oahOptionsVector.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff    972 Oct 15 20:16:51 2021 oahBasicTypes.h
64 -rw-r--r--@  1 jacquesmenu  staff  29773 Oct 15 18:48:10 2021 mfStringsHandling.cpp
16 -rw-r--r--@  1 jacquesmenu  staff   6269 Oct 15 18:55:46 2021 mfStringsHandling.h
16 -rw-r--r--@  1 jacquesmenu  staff   5028 Oct  7 20:03:27 2021 mfTiming.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff   3726 Oct  8 08:21:09 2021 mfTiming.h
```

The files *Elements.h/.cpp contain base classes to variants, such as src/formats/lpsr//lpsrElements.h/.cpp, whose lpsrElement class is used in a number of other files:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grep -r 'public lpsrElement' *
formats/lpsr/lpsrStaves.h:29:class EXP lpsrNewStaffGroupBlock : public lpsrElement
formats/lpsr/lpsrStaves.h:87:class EXP lpsrNewStaffTuningBlock : public lpsrElement
formats/lpsr/lpsrStaves.h:150:class EXP lpsrNewStaffBlock : public lpsrElement
formats/lpsr/lpsrStaves.h:208:class EXP lpsrStaffBlock : public lpsrElement
formats/lpsr/lpsrVariables.h:27:class EXP lpsrVariableUseCommand : public lpsrElement
formats/lpsr/lpsrScores.h:35:class EXP lpsrScore : public lpsrElement
formats/lpsr/lpsrBarNumbers.h:26:class EXP lpsrBarNumberCheck : public lpsrElement
formats/lpsr/lpsrBarNumbers.h:85:class EXP lpsrBarCommand : public lpsrElement
formats/lpsr/lpsrLyrics.h:31:class EXP lpsrNewLyricsBlock : public lpsrElement
formats/lpsr/lpsrComments.h:25:class EXP lpsrComment : public lpsrElement
formats/lpsr/lpsrVoices.h:29:class EXP lpsrUseVoiceCommand : public lpsrElement
formats/lpsr/lpsrParts.h:27:class EXP lpsrPartBlock : public lpsrElement
formats/lpsr/lpsrPapers.h:32:class EXP lpsrPaper : public lpsrElement
formats/lpsr/lpsrPartGroups.h:29:class EXP lpsrPartGroupBlock : public lpsrElement
formats/lpsr/lpsrParallelMusic.h:28:class EXP lpsrParallelMusicBLock : public lpsrElement
formats/lpsr/lpsrLayouts.h:23:class EXP lpsrLayout : public lpsrElement
formats/lpsr/lpsrHeaders.h:27:class EXP lpsrHeader : public lpsrElement
formats/lpsr/lpsrScheme.h:29:class EXP lpsrSchemeVariable : public lpsrElement
formats/lpsr/lpsrScheme.h:140:class EXP lpsrSchemeFunction : public lpsrElement
formats/lpsr/lpsrBookBlockElements.h:35:class EXP lpsrBookBlockElement : public
    lpsrElement
formats/lpsr/lpsrBookBlockElements.h:237:class EXP lpsrBookBlock : public lpsrElement
formats/lpsr/lpsrContexts.h:30:class EXP lpsrContext : public lpsrElement
```

There are a number of self-explaning `*BasicTypes.h/.cpp` file names:

```
./formats/msdl/msdlEnumTypes.h
./formats/msdl/msdlEnumTypes.cpp
./formats/msr/msrBasicTypes.cpp
./formats/msr/msrBasicTypes.h
./formats/lpsr/lpsrEnumTypes.cpp
./formats/lpsr/lpsrEnumTypes.h
./formats/bsr/bsrEnumTypes.h
./formats/bsr/bsrEnumTypes.cpp
./oah/oahBasicTypes.h
./oah/oahBasicTypes.cpp
./formatsgeneration/msrGeneration/msrGenerationBasicTypes.cpp
./formatsgeneration/msrGeneration/msrGenerationBasicTypes.h
```

The files are grouped in the `src` folder according to the component they belong to:

- `converters`
- `generators`
- `interfaces`
- `oah`
- `formatsgeneration`
- `passes`
- `formats`
- `utilities`
- `wae`

13.2 Adding C++ files

Building MusicFormats relies on `build/CMakeLists.txt` to find the C++ files that should be compiled.

When building MusicFormats with:

```
cd build
make
```

a `cmake` cache is created in file `build/libdir/CMakeCache.txt`, containing the list of all the C++ files in the library, including those of the embedded `libmusicxml2`.

Adding individual files in fine, but adding new folders in the `src` hierarchy implies to update variable `SRC_FOLDERS` in `build/CMakeLists.txt` accordingly and to remove the `build/libdir/CMakeCache.txt` cache.

Care must be taken when adding a new file on a case insensitive file system: the type case in its name should be what is needed in the first place.

For example, renaming `src/formats/msr/msrBarlines.h` to `src/formats/msr/msrBarLines.h` (this author's experience) leads MusicFormats not to build successfully on Linux if you develop on Windows or MacOS:

- the latter two usually use case insensitive file names (even though one may choose to format as disk to be case-sensitive), but Linux does not;

- the renaming above is not pushed to the repository by `git push` on case insensitive file systems.

The best solution here, both for files and folders names, is to use '`git mv`' to do the renaming instead of the operating system tools:

- `git mv oldName newName`
- `git commit "..." -a`
- `git push`

13.3 Renaming C++ files

Renaming a C++ file causes `build/libdir/CMakeCache.txt` to be obsolete: it then has to be removed, and the library should be built anew.

For example, this author uses the `rmcache` bash alias to remove the cache:

```
jacquesmenu@macmini: ~/musicformats-git-dev > type rmcache
rmcache is aliased to ‘rm /Users/jacquesmenu/musicformats-git-dev/build/libdir/CMakeCache.
  txt’
```

Running `make` will re-create this cache with the new file name.

Caution has to be taken when a file name case is changed in a case-insensitive development environment such as Windows or MacOS. Cloning MusicFormats in Linux will then fail to find the file under its new name.

In such a case, the following Git command has to be used to actually change the file name in MusicFormats repository:

```
git mv -f <old name> <new name>
```

Changing the name of a directory in `src/` should be propagated to `build/CMakeLists.txt`, since this is where the set of files to be compiled is determined, as in:

```
if (MSR)
  set (SRC_FOLDERS ${SRC_FOLDERS} passes/musicxml2mxsr formats/mxsr passes/mxsr2msr
    formats/msr passes/msr2msr formatsgeneration/multiGeneration formatsgeneration/
    msrGeneration generators/Mikrokosmos3Wandering generators/LilyPondIssue34)
endif()
```

In such a case, the cache should be removed before building, see `rmcache` above.

13.4 Source code layout

The following text-editing conventions are used:

- tabs are not used before the first non-space character in a line, two spaces are used instead;
- the code is not tightly packed: declarations in classes have the members' names aligned vertically, with many spaces before them if needed, and empty lines are used to separate successive activities in methods.

13.5 Defensive programming

The code base of `xml2ly` is *defensive programming* oriented, which means that:

- identifiers are explicit and long if needed – only very local ones are short, such as iteration loops indexes;
- the code is organized in sections, with an initial comment documenting what the code does;
- the C++17's `auto` declaration feature is used only for enumeration type `s`, see below. Writing the explicit types in a large code base helps the maintainer mastering the code;
- function `mfAssert ()` is used to perform sanity checks, such as detect a null pointer prior to using it.

The few uses of `auto` declarations are in range-based `for` loops over enumeration type `s`. There the type of the index is explicit from the `Enum*` being enumerated over. For example, in src/formats/msr/msrBasicTypes.cpp:

```
    for (auto e : mfEnumAll<msrHarmonyKind> ()) {
      // create the harmony structure
      S_msrHarmonyStructure
        harmonyStructure =
          msrHarmonyStructure::create (
            e);

      // register it in the map
      gGlobalHarmonyStructuresMap [e] =
        harmonyStructure;
    } // for
```

Class `mfEnumAll` is defined in src/mfutilities/mfEnumAll.h as:

```
template< typename T >
class   mfEnumAll
{
  public:

    class   Iterator
    {
      public:

        Iterator (int value)
          : fIterationIndex (value)
            {}

        T operator* (void) const
            { return (T) fIterationIndex; }

        void operator++ (void)
            {  ++fIterationIndex; }

        Bool operator != (Iterator rhs)
            {  return fIterationIndex != rhs.fIterationIndex; }

      private:

        int fIterationIndex;
     };
};
```

13.6 Sanity checks

They are performed to ensure that the formats in MusicFormats are consistent, to avoid ugly crashes. An example is:

```
    // get voice to insert harmonies into
    S_msrVoice
      voiceToInsertHarmoniesInto =
        fCurrentPart->
          getPartHarmoniesVoice ();

#ifdef MF_SANITY_CHECKS_ARE_ENABLED
    // sanity check
    mfAssert (
      __FILE__, __LINE__,
      voiceToInsertHarmoniesInto != nullptr,
      "voiceToInsertHarmoniesInto is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED
```

13.7 JMI comments

Comments containg `JMI` indicates that the code may have to be reconsidered in the future, should a problem arise. They are removed when it becomes obvious that the code is fine. JMI was the acronym for the author's activity as a software contractor long time ago.

13.8 Exported symbols

The classes and functions that need to be exported from the MusicFormats library in the Windows meaning are marked as such with an EXP specification:

```
class EXP smartable {
  // ... ... ...
};
```

```
EXP S_mxsrOahGroup createGlobalMxsrOahGroup ();
```

EXP is defined in libmusicxml/src/interface/exports.h and is non-blank only when building on Windows™:

```
#if defined(WIN32) // && !defined (GCC)

# ifdef MSVC
#  pragma warning (disable : 4267)
#  pragma warning (disable : 4275)
#  pragma warning (disable : 4251)
#  pragma warning (disable : 4786)
#  pragma warning (disable : 4251)
#  pragma warning (disable : 4275)
# endif

# ifdef LIBMUSICXML_EXPORTS
#  define EXP __declspec(dllexport)

# elif defined(LIBMUSICXML_STATIC)
#  define EXP

# else
#  define EXP __declspec(dllimport)
# endif

#else

# ifdef LIBMUSICXML_EXPORTS
#  define EXP __attribute__ ((visibility("default")))
# else
#  define EXP
# endif

#endif
```

13.9 Smart pointers

libmusicxml2 provides what Dominique Fober named smart pointers, because:

- a smart pointer is an instance of a class that contains the actual pointer in the usual C++ sense;
- the actual pointer is guaranteed to be initialized to nullptr;
- garbage collection is implicit, using reference counts.

The definitions are in libmusicxml/src/lib!smartpointer.h.

The reference counting is done in class smartable:

```
class EXP smartable {
  private:
    unsigned  refCount;
  public:
    //! gives the reference count of the object
    unsigned refs() const         { return refCount; }
    //! addReference increments the ref count and checks for refCount overflow
    void addReference()           { refCount++; assert(refCount != 0); }
    //! removeReference delete the object when refCount is zero
    void removeReference()      { if (--refCount == 0) delete this; }

  protected:
    smartable() : refCount(0) {}
    smartable(const smartable&): refCount(0) {}
    //! destructor checks for non-zero refCount
    virtual ~smartable()    { assert (refCount == 0); }
    smartable& operator=(const smartable&) { return *this; }
};

A smart pointer type is created with template class   {\tt SMARTP}, for example:
template<class   T> class   SMARTP {
class EXP msrElement : public smartable
{
  // ... ... ...
};
typedef SMARTP<msrElement> S_msrElement;
```

Smart pointer type name belonging to Dominique Fober's work can be told from those of MusicFormats by their prefix:

- in `libmusicxml2`, smart pointer type names start with an 'S', such as `$xmlelement`;
- in MusicFormats, they start with 'S_', such as `S_oahHandler`.

Inheriting from class `smartable` is used to create smart pointer types, as in `src/wae/waeHandlers.h`:

```
class EXP waeHandler : public smartable
{
  public:

    // creation
    // ------------------------------------------------------

    static SMARTP<waeHandler> create ();

  public:

    // constructors/destructor
    // ------------------------------------------------------

                              waeHandler ();

    virtual                   ~waeHandler ();

  public:

    // set and get
    // ------------------------------------------------------

  public:

    // public services
    // ------------------------------------------------------

  public:

    // print
    // ------------------------------------------------------

    std::string           asString () const;

    void                  print (std::ostream& os) const;

  private:

    // private fields
    // ------------------------------------------------------
};
typedef SMARTP<waeHandler> S_waeHandler;
EXP std::ostream& operator<< (std::ostream& os, const S_waeHandler& elt);
```

The creation of the instances in src/wae/waeHandlers.cpp is done with:

```
S_waeHandler waeHandler::create ()
{
  waeHandler* obj =
    new waeHandler ();
  assert (obj != nullptr);
  return obj;
}
```

13.10 Files contents layout

Indentation is done by two spaces, avoiding TAB characters.

In `*.h` files, the classes declarations contain all of part of the following:

- public data types, usually enumeration type `s`, if any;
- public static class `create* ()` methods, except for pure virtual classes, in which case they are commented out;
- constructors ad destructor;
- public `set* ()` and `get* ()` methods;
- public services if any;
- public visiting methods, i.e. `acceptIn ()`, `acceptOut ()` and `browseData ()`, if the class contains browsable data such as STL lists, vectors, maps and sets;
- public print methods, such as `asString ()` and `print ()`;
- private methods if any;
- private fields.
- private work methods if any;
- private work fields.

A work method is used internally by the class, while a work field is one that evolves as the class contents is populated.

Most class declarations are followed by a smart pointer type and a `operator <<`, such as:

```
typedef SMARTP<msrHarmonyDegree> S_msrHarmonyDegree;
EXP std::ostream& operator << (std::ostream& os, const S_msrHarmonyDegree& elt);
```

The same order for constructors, destructor and methods is followed in most `.cpp` files.

13.11 `#define DEBUG*` code sections

Some sections of code in `.cpp` are controlled by such definitions:

- `//#define DEBUG_EARLY_OPTIONS::in` `src/oah/oahEarlyOptions.cpp`;
- `//#define DEBUG_INDENTER::in` `src/oah/mfIndentedTextOutput.cpp`;
- `//#define DEBUG_SPLITTING::in` `src/mfutilities/mfStringsHandling.cpp`

These can be uncommented to obtain development-time trace information, without there being a need for such in MusicFormats library regular use.

13.12 Identifiers choice conventions

The following rules apply:

- all enumeration type names describing variants in classes end in '`Kind`';
- all enumeration constants start with '`k`'. common prefixes are used to help locate all occurences of constants of the given type in a text editor, as:

```
enum class msrBassFigurePrefixKind {
  kBassFigurePrefix_UNKNOWN_,
  kBassFigurePrefixDoubleFlat, kBassFigurePrefixFlat, kBassFigurePrefixFlat,
  kBassFigurePrefixNatural,
  kBassFigurePrefixSharpSharp, kBassFigurePrefixSharp, kBassFigurePrefixDoubleSharp
};
```

- all classes names have a prefix indicating which part of MusicFormats there belong to, such as class `msrTimeSignature`, `oahAtomStoringAValue` and `msdlKeywordsLanguageAtom`;
- all classes member fields start with '`f`';
- all class static fields start with '`s`';
- all global variables start with '`pGlobal`', in which case there are `set`... and `get`... macros to access them, starting with '`g`' – this is to facilitate typing and names completion in code editors;
- some (all? JMI) start with '`g`';
- all variables private to methods start with '`pPrivate`';
- some `K_*` constants are declared as static class constant members, such as in `src/formats/msr/msrParts.h`:

```
//______________________________________________________________________________
class EXP msrPart : public msrPartGroupElement
{
  public:

    // constants
    // ------------------------------------------------------

    static const int K_PART_HARMONIES_STAFF_NUMBER;
    static const int K_PART_HARMONIES_VOICE_NUMBER;

    static const int K_PART_FIGURED_BASS_STAFF_NUMBER;
    static const int K_PART_FIGURED_BASS_VOICE_NUMBER;

   // ... ... ...
};
```

with the definition in src/formats/msr/msrParts.cpp:

```
//______________________________________________________________________________
// constants
const int msrPart::K_PART_HARMONIES_STAFF_NUMBER = 10;
const int msrPart::K_PART_HARMONIES_VOICE_NUMBER = 11;

const int msrPart::K_PART_FIGURED_BASS_STAFF_NUMBER = 20;
const int msrPart::K_PART_FIGURED_BASS_VOICE_NUMBER = 21;
```

When a field is an STL container, such a `vector`, `list`, `map` or `set`, this in indicated as part of the identifier, such as:

```
    std::map<std::string, std::string>
                          fPartsRenamingMap;
```

or

```
                          fStringToDalSegnoKindMapVariable;
```

or

```
    std::map<std::string, Sxmlelement>  fPartMeasureNumbersToElementsMap;
```

All `create* ()` methods create class instances, and are paired with an explicit constructor with the same parameters:

```
    // creation from MusicXML
    // ------------------------------------------------------

    static SMARTP<msrHarmonyDegree> create (
                            int                       inputLineNumber,
                            int                       harmonyDegreeValue,
                            msrAlterationKind         harmonyDegreeAlterationKind,
                            msrHarmonyDegreeTypeKind  harmonyDegreeTypeKind);

  protected:

    // constructors/destructor
    // ------------------------------------------------------

                          msrHarmonyDegree (
                            int                       inputLineNumber,
                            int                       harmonyDegreeValue,
                            msrAlterationKind         harmonyDegreeAlterationKind,
                            msrHarmonyDegreeTypeKind  harmonyDegreeTypeKind);

    virtual               ~msrHarmonyDegree ();
```

Some classes use private fields and methods for their internal working. A field in the form `fCurrent*` denotes something whose value is not permanent once set. Fields named `fPending*` contain values gathered to be used later, such as `fPendingHarmoniesList` in `src/passes/mxsr2msr/mxsr2msrTranslator.h/.cpp`.

13.13 Exceptions and warnings/errors reporting

MusicFormats defines exceptions for its needs in `src/wae/waeExceptions.h/.cpp`. These exceptions can be related to a format, a pass or a converter. Exceptions named `*Internal*` are raised when something that should not happen occurs: this to avoid ugly crashes later in the execution.

All the std::exception classes are derived from , that contains:

```
class EXP mfException: public std::exception
{
  public:

    // constructors/destructor
    // ------------------------------------------------------

                              mfException (
                                std::string const& exceptionDescription = "",
                                int            exceptionNumber = 0,
                                int            exceptionLevel = 0
                              ) throw ()
                                : fExceptionDescription (
                                    "mfException: " + exceptionDescription),
                                  fExceptionNumber (exceptionNumber),
                                  fExceptionLevel (exceptionLevel)
                              {}

  // ... ... ...

  private:

    std::string               fExceptionDescription;

    int                       fExceptionNumber;
    int                       fExceptionLevel;
};
```

An example of std::exception is:

```
class EXP mxsr2msrException: public mfException
{
  public:
    mxsr2msrException (std::string const& exceptionDescription) throw ()
      : mfException (exceptionDescription)
    {}
};
typedef SMARTP<musicxmlException> S_musicxmlException;
```

There are warning and error reporting functions in `src/wae/waeInterface.h.h/.cpp`. Examples are:

```
void oahValueFittedAtom::applyValueLessAtom (std::ostream& os)
{
  std::stringstream ss;

  ss <<
    "Applying value-fitted atom '" <<
    fetchNames () <<
    "' without a value";

  oahInternalError (ss.str ());
}

  fSelected = true;
```

and:

```
    case msrPedalTypeKind::kPedalType_UNKNOWN_:
      {
        // should not occur

        std::stringstream ss;

        ss <<
          "msrPedal '" <<
          elt->asShortString () <<
          "' has no pedal type";

        msrInternalError (
          gServiceRunData->getInputSourceName (),
          inputLineNumber,
          __FILE__, __LINE__,
          ss.str ());
      }
      break;
```

Another one is:

```
void mxsr2msrTranslator::visitEnd ( S_accordion_registration& elt )
{
  int inputLineNumber =
    elt->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> End visiting S_accordion_registration" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // An accordion-registration element needs to have
  // at least one of the child elements present

  if (fCurrentAccordionNumbersCounter == 0) {
    musicxmlWarning (
      gServiceRunData->getInputSourceName (),
      inputLineNumber,
      "accordion-registration has 0 child element, ignoring it");
  }

  else {
    // create the accordion registration
    S_msrAccordionRegistration
      accordionRegistration =
        msrAccordionRegistration::create (
          inputLineNumber,
          fCurrentAccordionHigh,
          fCurrentAccordionMiddle,
          fCurrentAccordionLow);

    // append it to the current part
    fCurrentPart->
      appendAccordionRegistrationToPart (
        accordionRegistration);
  }
}
```

13.14 Exporting symbols for Windows DLLs

Windows needs export specifications for the symbols used by clients of a DLL.

https://docs.microsoft.com/en-us/cpp/build/exporting-from-a-dll-using-declspec-dllexport?view=msvc-160 is titled "Exporting from a DLL Using `__declspec(dllexport)`". It states that:

- to export functions, the `__declspec(dllexport)` keyword must appear to the left of the calling-convention keyword, if a keyword is specified. For example:

```
__declspec(dllexport) void __cdecl Function1(void);
```

- to export all of the public data members and member functions in a class, the keyword must appear to the left of the class name as follows:

```
class  __declspec(dllexport) CExampleExport : public CObject
{ ... class  definition ... };
```

MusicFormats uses symbol `EXP`, supplied by `libmusicxml/samples/`.
It is defined in `libmusicxml/src/elements!exports.h` to be empty except on Windows, where it is a default visibility attribute:

```
#ifndef __exports__
#define __exports__

#if defined(WIN32) // && !defined (GCC)

# ifdef MSVC
#  pragma warning (disable : 4267)
#  pragma warning (disable : 4275)
#  pragma warning (disable : 4251)
#  pragma warning (disable : 4786)
#  pragma warning (disable : 4251)
#  pragma warning (disable : 4275)
# endif

# ifdef LIBMUSICXML_EXPORTS
#  define EXP __declspec(dllexport)

# elif defined(LIBMUSICXML_STATIC)
#  define EXP

# else
#  define EXP __declspec(dllimport)
# endif

#else

# ifdef LIBMUSICXML_EXPORTS
#  define EXP __attribute__ ((visibility("default")))
# else
#  define EXP
# endif
```

```

#endif

#endif
```

13.15 Dynamic type checking

Enumeration type s are not ideal to distinguish variants when inheritance is used, mainly because adding new derived types imposes the addition of new constants, thus impacting other areas in the code base.

dynamic_cast is used in those cases, such as:

```
  // handle the option
  if (
    // options group?
    S_oahGroup
      group =
        dynamic_cast<oahGroup*>(&(*element))
  ) {
    registerOahElementUse (
      group, optionNameUsed, ""); // "===group==="); // JMI to debug
  }

  else if (
    // options subgroup?
    S_oahSubGroup
      subGroup =
        dynamic_cast<oahSubGroup*>(&(*element))
  ) {
    registerOahElementUse (
      subGroup, optionNameUsed, ""); // "===subGroup==="); // JMI to debug
  }
```

13.16 Input line numbers

The passes and converters in MusicFormats convert formats of scores from one format to another. In order to produce helpful warning and error messages, several descriptions contain a field:

```
    int                       fInputStartLineNumber;
```

An input line number in the xmlelement class is the only thing that has had to be added to libmusicxml2 for the needs of MusicFormats.

Also, many methods contain an int inputLineNumber parameter, which is always the first one:

```
msrElement::msrElement (
  int inputLineNumber)
{
  fInputStartLineNumber = inputLineNumber;
}
```

Such input line numbers can be present in the output of the converters, such as:

```
jacquesmenu@macmini: ~/musicformats-git-dev/files > xml2ly -query input-line-numbers
--- Help for atom "input-line-numbers" in subgroup "Output"
    -iln, -input-line-numbers
          Generate after each note and barLine a comment containing
          its MusicXML input line number.
          This is useful when debugging xml2ly.
```

Generators such as `Mikrokosmos3Wandering` don't read any input, and the input line numbers the use are the ones in the source code, which is helpful for debugging. For example:

```
S_msrStaff Mikrokosmos3WanderingGenerator::createStaffInPart (
  int               staffNumber,
  const S_msrPart& part)
{
  // create the staff
  S_msrStaff
    staff =
      msrStaff::create (
        __LINE__, msrStaffKind::kStaffKindRegular, staffNumber, part);

  // append it to the part
  part ->
    addStaffToPartCloneByItsNumber ( // JMI NOT clone???
      staff);

  return staff;
}
```

In all output produced by MusicFormats services, including trace informations and comments in the generated output refering to the input data, line numbers appear as:

```
  line <number>
```

with a single space before the number, as in:

```
  e16 %{ line 153 %} ] %{ line 163 kBeamEnd %}
```

This helps locating such occurrences in the debug process.

13.17 Static declarations

They are used for:

- classes methods such as method `msrTie::create ()`, method `msrTempo::createTempoPerMinute ()` and method `msrTemp::msrTempoBeatUnitsKindAsString ();`
- classes constant fields such as constant `msrStanza::K_STANZA_NUMBER_UNKNOWN_`, to be prefered to C-style `#define` preprocessor specifications for type safety;
- functions of methods remanent variables such as function private variable `pPrivateThisMethodHasBeenRun` in function `initializeMsrGenerationAPI ();`
- library-wide variables such as global variable `gGlobalOStreamIndenter` and global variable `sGlobalTimingItem` that would be too cumbersome to pass to each and every method or function that uses them.

13.18 Avoiding MusicFormats multiple initializations

Such behaviour would create data structures several times, the result being unnecessary activities being performed. Avoiding it is done with function private variable `pPrivateThisMethodHasBeenRun`, here in src/formats/bsr/bsr.cpp:

```
void initializeBSR ()
{
  // protect library against multiple initializations
  static Bool pPrivateThisMethodHasBeenRun (false);

  if (! pPrivateThisMethodHasBeenRun) {
#ifdef MF_TRACE_IS_ENABLED
    if (gEarlyOptions.getTraceEarlyOptions () && ! gEarlyOptions.getEarlyQuietOption ()) {
      gLog <<
        "Initializing BSR basic types handling";

      gWaeHandler->waeTraceWithoutInputLocation (
        __FILE__, __LINE__,
        ss.str ());
    }
#endif // MF_TRACE_IS_ENABLED

    // BSR braille output kinds handling
    // ------------------------------------------------------

    initializeBsrBrailleOutputKindsMap ();

    // BSR texts languages handling
    // ------------------------------------------------------

    initializeBsrTextsLanguageKindsMap ();

    pPrivateThisMethodHasBeenRun = true;
  }
}
```

13.19 Enumeration types

All enumeration type `s` use the C++17 '`enum class`' feature, such as:

```
enum class msrSlurTypeKind {
  kSlurType_UNKNOWN_,

  kSlurTypeRegularStart, kSlurTypePhrasingStart,
  kSlurTypeContinue,
  kSlurTypeRegularStop, kSlurTypePhrasingStop
};
```

This prevents enumeration constants name conflicts across enumeration type `s`, and qualified names such as constant `msrSlurTypeKind::kSlurTypeRegularStart` are quite explicit.

Many enumerations names end in '`Kind`', which is a way distinguish them from rather similar classes names in some cases.

Enumeration constants in the form `k*_UNKNOWN_` are used to indicate a value that *has not been set yet*. There are always the first on in the corresponding enumeration, to benefit from the the C++17 implicit initialization to the the equivalent of 0.

An enumeration constant may end in '`None`', meaning that `"none"` is actually a possible value for the corresponding type:

```
    // repeat winged
    enum class msrBarLineRepeatWingedKind {
      kBarLineRepeatWingedNone,

      kBarLineRepeatWingedStraight, kBarLineRepeatWingedCurved,
      kBarLineRepeatWingedDoubleStraight, kBarLineRepeatWingedDoubleCurved
    };
```

Here how the "winged" MusicXML attribute of <repeat/> is analysed in :

```
void mxsr2msrTranslator::visitStart ( S_repeat& elt )
{
    // ... ... ...

    std::string winged = elt->getAttributeValue ("winged");

    fCurrentBarLineRepeatWingedKind =
      msrBarLineRepeatWingedKind::kBarLineRepeatWingedNone; // default value

    if (winged.size ()) {
      if       (winged == "none") {
        fCurrentBarLineRepeatWingedKind =
          msrBarLineRepeatWingedKind::kBarLineRepeatWingedNone;
      }
      else if (winged == "straight") {
        fCurrentBarLineRepeatWingedKind =
          msrBarLineRepeatWingedKind::kBarLineRepeatWingedStraight;
      }
      else  if (winged == "curved") {
        fCurrentBarLineRepeatWingedKind =
          msrBarLineRepeatWingedKind::kBarLineRepeatWingedCurved;
      }
      else  if (winged == "double-straight") {
        fCurrentBarLineRepeatWingedKind =
          msrBarLineRepeatWingedKind::kBarLineRepeatWingedDoubleStraight;
      }
      else  if (winged == "double-curved") {
        fCurrentBarLineRepeatWingedKind =
          msrBarLineRepeatWingedKind::kBarLineRepeatWingedDoubleCurved;
      }
      else {
        std::stringstream ss;

        ss <<
          "repeat winged \"" << winged <<
          "\" is unknown";

        musicxmlError (
          gServiceRunData->getInputSourceName (),
          inputLineNumber,
          __FILE__, __LINE__,
          ss.str ());
      }
    }

    // ... ... ...
```

The MusicFormats enumeration type s come with a << operator to display their values in a user friendly way, such as type msrChordInKind in src/formats/msr/msrBasicTypes.h/.cpp:

```
enum class msrChordInKind {
  kChordIn_UNKNOWN_,

  kChordInMeasure,
  kChordInTuplet,
  kChordInGraceNotesGroup
};

EXP std::string msrChordInKindAsString (
  msrChordInKind chordInKind);

std::ostream& operator << (std::ostream& os, const msrChordInKind& elt);
```

The implementation is:

```
std::string msrChordInKindAsString (
  msrChordInKind chordInKind)
{
  std::string result;

  switch (chordInKind) {
    case msrChordInKind::kChordIn_UNKNOWN_:
      result = "kChordIn_UNKNOWN_";
      break;
    case msrChordInKind::kChordInMeasure:
      result = "kChordInMeasure";
      break;
    case msrChordInKind::kChordInTuplet:
      result = "kChordInTuplet";
      break;
    case msrChordInKind::kChordInGraceNotesGroup:
      result = "kChordInGraceNotesGroup";
      break;
  } // switch

  return result;
}

std::ostream& operator << (std::ostream& os, const msrChordInKind& elt)
{
  os << msrChordInKindAsString (elt);
  return os;
}
```

13.20 yes/no enumerations types

Boolean argument to methods calls are fine in simple cases such as:

```
    void                      setCombinedBooleanVariables (Bool value);
```

But when there are multiple arguments, the semantics of the `true` or `false` constants if far from obvious.

This is why we use enum classes such as:

```
enum class msrVoiceCreateInitialLastSegmentKind {
  kCreateInitialLastSegmentYes,
  kCreateInitialLastSegmentNo
};
```

in such cases, so that the arguments bare a clear semantics:

```
  fPartHarmoniesVoice =
    msrVoice::create (
      inputLineNumber,
      msrVoiceKind::kVoiceKindHarmonies,
      partHarmoniesVoiceNumber,
      msrVoiceCreateInitialLastSegmentKind::kCreateInitialLastSegmentYes,
      fPartHarmoniesStaff);
```

13.21 Boolean values anyway

Defining a yes/no enumeration type for 'true' boolean values such as the variables containing the OAH options would be cumbersome. The C++ `bool` type suffers from the C heritage, in which integers and even pointers can be mixed in and considered as boolean values.

Moreover, a `bool` variable not explicitly initialized in the developper's code can lead to hard to fix bugs, in particular when the MusicFormats library is used on various hardware and operating systems.

For these reasons, MusicFormats features a class `Bool` defined in `src/utilities/mfBool.h/.cpp`. It encapsulates the actual `bool` value, enforcing that its initial value is not random, but known to the developper, through constructors. This also avoids in particular long sequences of initializations in the passes constructors.

13.22 On/off values

Some elements in MusicFormats are represented by a *three-state* value.

On/off values are handled by enumeration type `mfOnOffKind`, declared in `src/mflibrarymfOnOff.h`:

```
//______________________________________________________________________________
enum class mfOnOffKind {
  kMfOnOffUnknown,
  kMfOnOffOn, kMfOnOffOff
};

Bool mfOnOffKindAsBool (
  mfOnOffKind onOffKind);

std::string mfOnOffKindAsString (
  mfOnOffKind onOffKind);

std::ostream& operator << (std::ostream& os, const mfOnOffKind elt);
```

This is the case for example of page ragging information in `src/formats/lpsr//lpsr/lpsrPapers.h`:

```
    // on/off values
    mfOnOffKind           fRaggedBottom;
    S_oahOnOffAtom        fRaggedBottomAtom;

    mfOnOffKind           fRaggedLast;
    S_oahOnOffAtom        fRaggedLastAtom;

    mfOnOffKind           fRaggedLastBottom;
    S_oahOnOffAtom        fRaggedLastBottomAtom;

    mfOnOffKind           fRaggedRight;
    S_oahOnOffAtom        fRaggedRightAtom;
```

Care must be taken in `src/mflibrarymfOnOff.cpp` when converting an enumeration type `mfOnOffKind` value to type `Bool`:

```
Bool mfOnOffKindAsBool (
  mfOnOffKind onOffKind)
{
  Bool result;

  switch (onOffKind) {
    case mfOnOffKind::kMfOnOffUnknown:
      {
        mfError (
          __FILE__, __LINE__,
          "mfOnOffKind::kMfOnOffUnknown cannot be converted to Bool");
      }
      break;
    case mfOnOffKind::kMfOnOffOn:
      result = true;
      break;
    case mfOnOffKind::kMfOnOffOff:
      result = false;
      break;
  } // switch

  return result;
}
```

13.23 Iterating over numeration types

Such iterations rely on template classes. For this to work, the enumeration type should provide specific `AllFirst` and `AllLast` 'aliases' for the first and last constants in the type.

This is done for enumeration type `msrHarmonyKind` in `src/formats/msr/msrBasicTypes.h`:

```
// harmonies
//______________________________________________________________________________
enum class msrHarmonyKind {
  kHarmony_UNKNOWN_,

  // MusicXML harmonies

  kHarmonyMajor, kHarmonyMinor,
  kHarmonyAugmented, kHarmonyDiminished,

  // ... ... ...

  // other

  kHarmonyOther,

  kHarmonyNone,

  // aliases
  // ------------------------------------

  AllFirst = kHarmony_UNKNOWN_,
  AllLast  = kHarmonyNone,

  // ... ... ...
};
```

class `mfEnumAll` is defined in `src/mfutilities/mfEnumAll.h`:

```
//______________________________________________________________________________
/*
  https://stackoverflow.com/questions/8498300/allow-for-range-based-for-with-enum-classes
*/
template< typename T >
class   mfEnumAll
{
  public:

    class   Iterator
    {
      public:

        Iterator (int value)
          : fIterationIndex (value)
            {}

        T operator* (void) const
            { return (T) fIterationIndex; }

        void operator++ (void)
            {  ++fIterationIndex; }

        Bool operator != (Iterator rhs)
            {  return fIterationIndex != rhs.fIterationIndex; }

      private:

        int fIterationIndex;
    };
};

template< typename T >
typename mfEnumAll<T>::Iterator begin (mfEnumAll<T>)
{
  return typename mfEnumAll<T>::Iterator ((int)T::AllFirst);
}

template< typename T >
typename mfEnumAll<T>::Iterator end (mfEnumAll<T>)
{
  return typename mfEnumAll<T>::Iterator (((int)T::AllLast) + 1);
}
```

The `mfEnumAll` template class, defined in src/mfutilities/mfEnumAll.h can then be used to iterate from constant `msrHarmonyKind::AllFirst` to constant `msrHarmonyKind::AllLast`, here in src/formats/msr/msrBasicT

```
void initializeHarmonyStructuresMap ()
{
  // protect library against multiple initializations
  static Bool pPrivateThisMethodHasBeenRun (false);

  if (! pPrivateThisMethodHasBeenRun) {
    for (auto e : mfEnumAll<msrHarmonyKind> ()) {
      // create the harmony structure
      S_msrHarmonyStructure
        harmonyStructure =
          msrHarmonyStructure::create (
            e);

      // register it in the map
      gGlobalHarmonyStructuresMap [e] =
        harmonyStructure;
    } // for

    pPrivateThisMethodHasBeenRun = true;
  }
}
```

The `mfEnumAll` template class, defined in src/mfutilities/mfEnumAll.h can then be used to iterate from constant `msdlTokenKind::AllFirst` to constant `msdlTokenKind::AllLast`, here in src/formats/msdl/msdlTokens

```
    for (auto e : EnumNonSeparators<msdlTokenKind> ()) {
      std::string
        nonSeparatorTokenAsMsdlString  =
          msdlTokenKindAsMsdlString (
            e,
            keywordsLanguageKind);

      // ... ... ...
    } // for
```

All such class `Enum*` classes in MusicFormats are:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grep -r 'class   Enum' *
formats/msdl/msdlTokens.h:class   EnumNonSeparators
formats/msdl/msdlTokens.h:class   EnumLanguageIndependent
formats/msdl/msdlTokens.h:class   EnumLanguageDependent
formats/msr/msrBasicTypes.h:class   EnumTrueHarmonies
utilities/mfutilities.h:class   mfEnumAll
```

For example class `EnumTrueHarmonies`, that relies on constant `msrHarmonyKind::TrueHarmoniesFirst` and constant `msrHarmonyKind::TrueHarmoniesLast`:

```
void msrHarmonyStructure::printAllHarmoniesStructures (std::ostream& os)
{
  os <<
    "All the known harmonies structures are:" <<
    std::endl << std::endl;

  ++gIndenter;

  for (auto e : EnumTrueHarmonies<msrHarmonyKind> ()) {
    // create the harmony intervals
    S_msrHarmonyStructure
      harmonyStructure =
        msrHarmonyStructure::create (
          e);

    // print it
    os <<
      harmonyStructure <<
      std::endl;
  } // for

  --gIndenter;
}
```

13.24 mfRational numbers

MusicFormats uses rationals for notes sounding and display whole notes and measure positionss, among others. Class `mfRational` is defined by `libmusicxml2` in `libmusicxml/src/lib!mfRational.h/.cpp`:

```
class EXP mfRational {
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    long int              fNumerator;
    long int              fDenominator;

    // used by rationalise()
//    long int              gcd(long int a, long int b);

  // ... ... ...
};
```

Rationals are not used, however, for tuplets factors, see .

13.25 Don't mix pure and typed scalar types

A whole notes value is described by class `msrWholeNotes`, defined in `src/formats/msr/msrNotesDurations.h/.cpp`. It contains a numerator and denominator, as class `msrRational` does, but those types are distinct, since an `msrWholeNotes` instance is a fraction of a whole note, note a mere typeless fraction.

MusicFormats uses `msrWholeNotes` instances for notes sounding and display whole notes values;

For the same reason, class `msrTupletFactor`, defined in `src/formats/msr/msrTupletFactors.h/.cpp`, is used with specifically named fields:

```
class EXP msrTupletFactor
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    int                       fTupletActualNotes;
    int                       fTupletNormalNotes;
};
```

13.26 Default values

The guide lines for MusicFormats in this matter are:

- smart pointers are initialized to `nullptr` in the class `SMARTP` constructor (they're smart after all), defined by `libmusicxml2` in `libmusicxml/src/lib!smartpointer.h`:

```
template<class   T> class   SMARTP {
  private:
    //! the actual pointer to the class
    T* fSmartPtr;

  public:
    //! an empty constructor - points to null
    SMARTP()   : fSmartPtr(0) {}

  // ... ... ...
```

- all variables and classes fields of non-class types, such as `int`, `float` and enumeration type `s`, are to be initialized explicitly;
- MusicFormats functions and methods parameters never have default values: overloading is used instead.

13.27 create* methods

All concrete classes, i.e. those that are not pure virtual, have `create* ()` methods paired with a constructor with the exact same parameters.

In most cases, there are justs named `create* ()`, but a couple of them have more explicit names.

One case is that of class `msrTempo` in `src/formats/msr/msrTempos.h/.cpp`, because calls to them would be hard to distinguish at first glance otherwise:

```
 class EXP msrTempo : public msrMeasureElement
{
    // ... ... ...

    static SMARTP<msrTempo> createTempoWordsOnly (
      int                  inputLineNumber,
      S_msrWords           tempoWords);

    static SMARTP<msrTempo> createTempoPerMinute (
      int                  inputLineNumber,
      msrDottedNotesDuration tempoBeatUnit,
      std::string          tempoPerMinute,
      msrTempoParenthesizedKind
                           tempoParenthesizedKind,
      msrPlacementKind     tempoPlacementKind);

    static SMARTP<msrTempo> createTempoBeatUnitEquivalent (
      int                  inputLineNumber,
      msrDottedNotesDuration tempoBeatUnit,
      msrDottedNotesDuration tempoEquivalentBeatUnit,
      msrTempoParenthesizedKind
                           tempoParenthesizedKind,
      msrPlacementKind     tempoPlacementKind);

    static SMARTP<msrTempo> createTempoNotesRelationship (
      int                  inputLineNumber,
      S_msrTempoNotesRelationshipElements
                           tempoNotesRelationshipLeftElements,
      msrTempoNotesRelationshipKind
                           tempoNotesRelationshipKind,
      S_msrTempoNotesRelationshipElements
                           tempoNotesRelationshipRightElements,
      msrTempoParenthesizedKind
                           tempoParenthesizedKind,
      msrPlacementKind     tempoPlacementKind);

    // ... ... ...
};
```

Another case is that of class `msrKey` in `src/formats/msr/msrKeys.h/.cpp`, in which the variant selected is made explicit:

```
class EXP msrKey : public msrMeasureElement
{
    // ... ... ...

    static SMARTP<msrKey> createTraditional (
                            int                       inputLineNumber,
                            msrQuarterTonesPitchKind keyTonicPitchKind,
                            msrModeKind               modeKind,
                            int                       keyCancel);

    static SMARTP<msrKey> createHumdrumScot (
      int                   inputLineNumber);

    // ... ... ...
};
```

13.28 get*(), set*() and fetch*() methods

As is usual, classes privates member variables are accessed through `set*()` () and `get*()` () methods. The name of these methods is obtained by replacing the '`f`' in the field name by '`set`' and '`get`', respectively. In `src/formats/msr/msrTies.h`, one finds:

```
    // set and get
    // ------------------------------------------------------

    void                      setTieKind (msrTieKind tieKind)
                                  { fTieKind = tieKind; }

    msrTieKind                getTieKind () const
                                  { return fTieKind; }

    void                      setTiePlacementKind (msrPlacementKind placementKind)
                                  { fTiePlacementKind = placementKind; }

    msrPlacementKind          getTiePlacementKind () const
                                  { return fTiePlacementKind; }
```

`fetch` is used when the result is not store in a variable, but has to computed in some way. `src/formats/msr/msrSegme`
contains:

```
S_msrStaff msrSegment::fetchSegmentUpLinkToStaff () const
{
  S_msrStaff result;

  if (fSegmentUpLinkToVoice) {
    result =
      fSegmentUpLinkToVoice->
        getVoiceUpLinkToStaff ();
  }

  return result;
}
```

Some methods exist in two versions, the second one with a `NonConst` suffix in the name. This can be the case if a private class field is to be modified outside the class, or if there are specific needs.

This happens for example in `src/formats/msr/mfslNotes.h/.cpp`:

```
    // articulations
    const std::list<S_msrArticulation>&
                              getNoteArticulationsList () const
                                  { return fNoteArticulationsList; }

    std::list<S_msrArticulation>&
                              getNoteArticulationsListNonConst ()
                                  { return fNoteArticulationsList; }
```

Another case is in `src/interpreters/mfsl/mfslDriver.h/.cpp`:

```
    const yy::location&   getScannerLocation () const
                            { return fScannerLocation; }

    yy::location&         getScannerLocationNonConst ()
                            // no const here
                            // due to constraints in the Flex-generated code
                            { return fScannerLocation; }
```

13.29 initialize*() and finalize*() methods

When a description contains many fields, the ones initialized by the values of the constructor's parameters are initialized in the latter, and the others are in an `initialize*()` `()` method, such as:

```
msrPart::msrPart (
  int               inputLineNumber,
  std::string         partID,
  S_msrPartGroup partUpLinkToPartGroup)
    : msrPartGroupElement (inputLineNumber)
{
  // replace spaces in partID to set fPartID
  for_each (
    partID.begin (),
    partID.end (),
    mfStringSpaceReplacer (fPartID, '_'));

/* JMI
#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  mfAssert (
    __FILE__, __LINE__,
    partUpLinkToPartGroup != nullptr,
    "partUpLinkToPartGroup is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  // set part number
  fPartAbsoluteNumber = ++sPartsCounter;

  // set part's part group upLink
  fPartUpLinkToPartGroup = partUpLinkToPartGroup;

  // do other initializations
  initializePart ();
}
```

Some `finalize*()` `()` methods exist.

13.30 *asString() and *fromString() functions

Each enumeration type comes with an `*AsString()` `()` function, to display the constant values as strings. Some also have a `fromString` `()` function to convert strings to the corresponding constant. For example, one finds in `src/formats/msr/msrBasicTypes.h/.cpp`:

```
// placement
//______________________________________________________________________________
enum class msrPlacementKind {
  kPlacement_UNKNOWN_,

  kPlacementAbove, kPlacementBelow
};
```

```
// placement
//______________________________________________________________________________
msrPlacementKind msrPlacementKindFromString (
  int    inputLineNumber,
  std::string placementString)
{
  msrPlacementKind result = msrPlacementKind::kPlacement_UNKNOWN_; // default value

  if      (placementString == "above")
    result = msrPlacementKind::kPlacementAbove;
  else if (placementString == "below")
    result = msrPlacementKind::kPlacementBelow;
  else {
    if (placementString.size ()) {
      std::stringstream ss;

      ss <<
        "placement \"" << placementString <<
        "\" should be 'above' or 'below'";

      musicxmlError (
        gServiceRunData->getInputSourceName (),
        inputLineNumber,
        __FILE__, __LINE__,
        ss.str ());
    }
  }

  return result;
}
```

```
std::string msrPlacementKindAsString (
  msrPlacementKind placementKind)
{
  std::string result;

  switch (placementKind) {
    case msrPlacementKind::kPlacement_UNKNOWN_:
      result = "noPlacement";
      break;
    case msrPlacementKind::kPlacementAbove:
      result = "placementAbove";
      break;
    case msrPlacementKind::kPlacementBelow:
      result = "placementBelow";
      break;
  } // switch

  return result;
}
```

Many classes have `asStringShort()` `()` methods to provide more compact a description as the one provided by the corresponding `asString()` `()` method.

13.31 `translate*()` methods and `convert*()` functions

To translate and to convert are aliass in the context of MusicFormats.

For semantic clearness, `translate*()` `()` methods are supplied by the individual translators, as in `src/passes/msr2m`

```
//______________________________________________________________________________
class EXP msr2msrTranslator :

  // MSR score

  public visitor<S_msrScore>,

  // ... ... ...

{
  public:

                          msr2msrTranslator ();

    virtual               ~msr2msrTranslator ();

    S_msrScore            translateMsrToMsr (
                            const S_msrScore& theMsrScore);
  // ... ... ...

};
```

```
S_msrScore msr2msrTranslator::translateMsrToMsr (
  const S_msrScore& theMsrScore)
{
#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  mfAssert (
    __FILE__, __LINE__,
    theMsrScore != nullptr,
    "theMsrScore is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  // the MSR score we're visiting
  fVisitedMsrScore = theMsrScore;

  // create the resulting MSR score
  fResultingNewMsrScore =
    msrScore::create (
      K_MF_INPUT_LINE_UNKNOWN_,
      "msrScore::create()");

  // create a msrScore browser
  msrBrowser<msrScore> browser (this);

  // set the parts browsing order
  theMsrScore->
    setStavesBrowingOrderKind (
      msrStavesBrowingOrderKind::kStavesBrowingOrderHarmoniesFiguredBassesRegulars);

  // browse the visited MSR score
  browser.browse (*fVisitedMsrScore);

  // forget about the visited MSR score
  fVisitedMsrScore = nullptr;

  return fResultingNewMsrScore;
}
```

The convert*() () functions are the interfaces to the translators, for example in src/passes/msr2msr/msr2msrInte

```
S_msrScore translateMsrToMsr (
  S_msrScore        originalMsrScore,
  const S_msrOahGroup&     msrOpts,
  S_msr2msrOahGroup msr2msrOpts,
  mfPassIDKind           passIDKind,
  const std::string&      passDescription)
{
  // ... ... ...

  // the msr2msrTranslator
  msr2msrTranslator
    translator;

  // build the resulting MSR score
  S_msrScore
    resultingNewMsrScore =
      translator.translateMsrToMsr (
        originalMsrScore);

  // ... ... ...
}
```

13.32 context arguments

Some methods have such an argument, a std::string, to provide helpful information to the maintainer of MusicFormats. An exemple is method `msrMeasureRepeat::displayMeasureRepeat ()`, defined in src/formats/msr/ms

```
void msrMeasureRepeat::displayMeasureRepeat (
  int     inputLineNumber,
  const std::string& context)
{
  gLog <<
    std::endl <<
    "*********>> MeasureRepeat " <<
    ", measureRepeatMeasuresNumber: '" <<
    fMeasureRepeatMeasuresNumber <<
    ", measureRepeatSlashesNumber: '" <<
    fMeasureRepeatSlashesNumber <<
    "', voice:" <<
    std::endl <<
    fUpLinkToMeasureRepeatToVoice->getVoiceName () <<
    " (" << context << ")" <<
    ", line " << inputLineNumber <<
    " contains:" <<
    std::endl;

  ++gIndenter;
  print (gLog);
  --gIndenter;

  gLog <<
    " <<*********" <<
    std::endl << std::endl;
```

An call example in src/formats/msr/msrVoices.h is:

```
    displayVoiceMeasureRepeatAndVoice (
      inputLineNumber,
      "createMeasureRepeatFromItsFirstMeasures() 1");
```

13.33 Sorting and compare*() methods

MusicFormats sometimes needs to sort some data structures:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grep -r '\.sort (' *
oah/oahBasicTypes.cpp:  optionsMapElementsNamesList.sort ();
passes/lpsr2lilypond/lpsr2lilypondTranslator.cpp:    frameFrameNotesList.sort (
formats/msr/msrMeasuresSlices.cpp:  fSliceNotesFlatList.sort (
formats/msr/msrMeasuresSlices.cpp:  fSliceNoteEventsList.sort (
formats/msr/msrStaves.cpp:        fStaffAllVoicesList.sort (
formats/msr/msrStaves.cpp:        fStaffAllVoicesList.sort (
formats/msr/msrStaves.cpp:  fStaffRegularVoicesList.sort (
formats/msr/msrMeasures.cpp:    fMeasureElementsList.sort (
formats/msr/msrMeasures.cpp:    fMeasureElementsList.sort (
formats/msr/msrParts.cpp:    fPartAllStavesList.sort (
formats/msr/msrParts.cpp:    fPartAllStavesList.sort (
formats/lpsr/lpsrParts.cpp:    fPartBlockElementsList.sort (
```

There are thus a number of compare* () methods according to the needs:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grep -r compare * | grep '\.h'
oah/oahBasicTypes.h:    const std::multiset<S_oahElement, compareOahElements>&
oah/oahBasicTypes.h:    std::multiset<S_oahElement, compareOahElements>
oah/oahElements.h:struct compareOahElements {
formats/msr/msrParts.h:    static bool
    compareStavesToHaveFiguredBassesBelowCorrespondingPart (
formats/msr/msrNotes.h:    static bool           compareNotesByIncreasingMeasurePosition (
formats/msr/msrMeasureElementsWithoutUpLinkToMeasure.h:    static bool
    compareMeasureElementsByIncreasingMeasurePosition (
formats/msr/msrStaves.h:    static bool           compareVoicesByIncreasingNumber (
formats/msr/msrStaves.h:    static bool
    compareVoicesToHaveHarmoniesAboveCorrespondingVoice (
formats/msr/msrStaves.h:    static bool
    compareVoicesToHaveFiguredBassesBelowCorrespondingVoice (
formats/msr/msrMeasuresSlices.h:    static bool
    compareNotesEventsByIncreasingMeasurePosition (
formats/msr/msrMeasuresSlices.h:    static bool
    compareSimultaneousNotesChunksByIncreasingMeasurePosition (
formats/lpsr/lpsrParts.h:    static bool
    compareElementsToHaveHarmoniesAboveCorrespondingStaff (
formats/lpsr/lpsrParts.h:    static bool           compareStaffBlockWithOtherElement (
formats/lpsr/lpsrParts.h:    static bool
    compareChordNamesContextWithOtherElement (
utilities/mfutilities.h:    // compare indentation value
```

An example is:

```
bool msrPart::compareStavesToHaveFiguredBassesBelowCorrespondingPart (
  const S_msrStaff& first,
  const S_msrStaff& second)
{
  int
    firstStaffNumber =
      first->getStaffNumber (),
    secondStaffNumber =
      second->getStaffNumber ();

  if (firstStaffNumber > msrPart::K_PART_FIGURED_BASS_STAFF_NUMBER) {
    firstStaffNumber -= msrPart::K_PART_FIGURED_BASS_STAFF_NUMBER + 1;
  }
  if (secondStaffNumber > msrPart::K_PART_FIGURED_BASS_STAFF_NUMBER) {
    secondStaffNumber -= msrPart::K_PART_FIGURED_BASS_STAFF_NUMBER + 1;
  }

  bool result =
    firstStaffNumber > secondStaffNumber;

  return result;
}
```

13.34 Mutually dependent classes

C++17 has some constraints on how such classes can be defined, see for example https://cplusplus.com/forum/articles/10627/. MusicFormats sticks to having one header file per important class, with one exception.

std::endl;

In some cases, two classes have to know about each other, such as `msrPartGroup` and `msrPartGroupElement`. This is because part groups can be nested: a part group contains part group element, which can be staves as well as other part groups. This loop can be seen in red in figure 20.3 [The MSR classes hierarchy], page 199.

13.34.1 Pre-declarations

Handling such a mutual dependency in C++ is rather easy if both classes are declared in one and the same `.h` file, with a pre-declaration of one of them before the other one is declared, such as:

```
class   msrPartGroup;
typedef SMARTP<msrPartGroup> S_msrPartGroup;
```

All one can do with a forward declared type is to declare a *pointer* or *reference* to said type. In particular, this precludes a forward declared type to be that of a class member or the return type of a class method.

As a matter of taste, MusicFormats follows `libmusicxml2` by declaring the `create ()` static class methods this way:

```
//______________________________________________________________________________
class EXP msrBarLine : public msrMeasureElement
{
  public:

    // creation from MusicXML
    // ------------------------------------------------------

    static SMARTP<msrBarLine> create (
```

As an alternative, the same could be written as follows, at the cost of an extra pre-declaration for each class:

```
class EXP msrBarLine;
typedef   SMARTP<msrBarLine> S_msrBarLine;

class EXP msrBarLine : public msrMeasureElement
{
  public:

    // creation from MusicXML
    // ------------------------------------------------------

    static S_msrBarLine create (...)

  // ... ... ...
};
```

There are many classes pre-declations in MusicFormats. We thus use a couple of `*EnumTypes*` files to contain most of them, in order to avoid numerous header files includes:

```
jacquesmenu@macstudio-1:~/JMI_Developpement/musicformats-git-dev/src > look EnumTypes
./representations/msdl/msdlEnumTypes.h
./representations/msdl/msdlEnumTypes.cpp
./representations/msr/msrTupletsEnumTypes.h
./representations/msr/msrTemposEnumTypes.h
./representations/msr/msrTupletsEnumTypes.cpp
./representations/msr/msrMeasuresEnumTypes.cpp
./representations/msr/msrTemposEnumTypes.cpp
./representations/msr/msrStavesEnumTypes.h
./representations/msr/msrNotesEnumTypes.h
./representations/msr/msrNotesEnumTypes.cpp
./representations/msr/msrStavesEnumTypes.cpp
./representations/msr/msrMeasuresEnumTypes.h
./representations/msr/msrRepeatsEnumTypes.cpp
./representations/msr/msrRepeatsEnumTypes.h
./representations/lpsr/lpsrEnumTypes.cpp
./representations/lpsr/lpsrEnumTypes.h
./representations/bsr/bsrEnumTypes.cpp
./representations/bsr/bsrEnumTypes.h
```

13.34.2 Simple mutual dependency using separate header files

A `msrPartGroupElement` is either a `msrPart` or another `msrPartGroupElement`, since the latter can be nested.

A `msrPartGroupElement` cannot contain an uplink to an instance of `S_msrPartGroup`: pre-declaring type `S_msrPartGroup`:

```
class   msrPartGroup;
typedef SMARTP<msrPartGroup> S_msrPartGroup;
```

and using it as the type of a class member common to all `msrPartGroupElement` sub-classes leads to the following error:

```
error: member access into incomplete type 'MusicFormats::msrPartGroup'
```

So src/formats/msr/msrPartGroupsElements.h contains only:

```
/*
  Parts and part groups can be found in part groups,
  hence class   msrPartGroupElement
*/

class EXP msrPartGroupElement : public msrElement
/*
  a purely virtual common ancestor to the msrPartGroup and msrPart classes,
  which can be inside an msrPartGroup
*/
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    /*
      The part group uplink is declared in the sub-classes,
      i.e. msrPart and msrPartGroup,
      to allow for separate *.h files, C++ constraint
    */
};
```

Then a `msrPartGroup`, a sub-class of `msrPartGroupElement` declared in src/formats/msr/msrPartGroups.h, can contain such an uplink to a `msrPartGroup` instance:

```
#include "msrPartGroupElements.h"

// ... ... ...

class   msrPartGroup;
typedef SMARTP<msrPartGroup> S_msrPartGroup;

// ... ... ...

class EXP msrPartGroup : public msrPartGroupElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks

    S_msrPartGroup          fPartGroupUpLinkToPartGroup;
                              // part groups can be nested

    // ... ... ...

    // allowing for both parts and nested part groups as elements

    std::list<S_msrPartGroupElement>
                            fPartGroupElementsList;
};
```

Type `S_msrPartGroup` is used in the declaration of field `fPartGroupElementsList`, hence its pre-declaration.

Class `msrPart`, another sub-classof `msrPartGroupElement`, in defined in src/formats/msr/msrParts.h this way:

```
#include "msrPartGroupElements.h"

// ... ... ...

class EXP msrPart : public msrPartGroupElement
{
  // ... ... ...

  public:

    // set and get
    // ------------------------------------------------------

    // upLinks

    void                    setPartUpLinkToPartGroup (
                              const S_msrPartGroup& partGroup)
                                { fPartUpLinkToPartGroup = partGroup; }

    S_msrPartGroup          getPartUpLinkToPartGroup () const
                                { return fPartUpLinkToPartGroup; }
    // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks

    S_msrPartGroup        fPartUpLinkToPartGroup;

    // ... ... ...
};
```

This rather complex situation is depicted at the top of figure 20.3 [The MSR classes hierarchy], page 199.

13.34.3 More complex mutual dependencies

The case of notes, chords, tuplets and grace notes groups is more intricate:

- a note can be standalone in a measure;
- a note can be part of:
  - a chord;
  - a tuplet;
  - a grace notes group;
  - a double tremolo;
- a chord can be standalone in a measure;
- a chord can be part of:
  - a tuplet;
  - a grace notes group;
- a tuplet can be standalone in a measure;
- a tuplet can be part of:
  - another tuplet;
- a grace notes group is attached to:
  - a note;
- a double tremolo is standalone in a measure.

Class `msrDoubleTremolo` is a sub-class of `msrMeasureElement`.

Regarding classes `msrNote`, `msrChord` and `msrTuplet`:

- they have to be sub-classes of class `msrMeasureElement` in some way, since they can be standalone in a `msrMeasure` instance;
- they should be sub-classes of `msrTupletElement`, since they can be members of a `msrTuplet` instance.

This leads to the following hierarchy:

- `msrNote`, `msrChord` and `msrTuplet` are direct sub-classes of class `msrTupletElement`
- class `msrTupletElement` is a direct sub-class of `msrMeasureElement`.

This rather complex situation is depicted at the bottom of figure 20.3 [The MSR classes hierarchy], page 199.

But then, the mutual dependency of need a more complex representation.

13.35 Templates and functional programming usage

There are currently few templates in the MusicFormats code base, namely:

- some are used by the two-phase visitors pattern, see chapter 15 [The two-phase visitors pattern], page 102;
- some exist for enumeration types, such as:

```
template< typename T >
class   EnumNonSeparators
{
  public:

    class   Iterator
    {
      public:

        Iterator (int value)
          : fIterationIndex (value)
            {}

        T operator* (void) const
            { return (T) fIterationIndex; }

        void operator++ (void)
            {  ++fIterationIndex; }

        Bool operator != (Iterator rhs)
            {  return fIterationIndex != rhs.fIterationIndex; }

      private:

        int fIterationIndex;
     };
};

template< typename T >
typename EnumNonSeparators<T>::Iterator begin (EnumNonSeparators<T>)
{
  return typename EnumNonSeparators<T>::Iterator ((int)T::NonSeparatorsFirst);
}

template< typename T >
typename EnumNonSeparators<T>::Iterator end (EnumNonSeparators<T>)
{
  return typename EnumNonSeparators<T>::Iterator (((int)T::NonSeparatorsLast) + 1);
}
```

- some are used by the code created by `bison`, like:

```
    /// Construct and fill.
    template <typename T>
    value_type (YY_RVREF (T) t)
      : yytypeid_ (&typeid (T))
    {
      ISCM_ASSERT (sizeof (T) <= size);
      new (yyas_<T> ()) T (YY_MOVE (t));
    }
```

There could be more templates use once MusicFormats reaches a rather stable code base and it is clear what parts of it can be restructured with generic code.

In the same vein, there is little use as of this writing of higher-level facilities such as `lambda` and functors.

Part III

Maintainance tools

Chapter 14

Maintainance tools

They are located in devtools/ and meant to facilitate the MusicFormats maintainance activities:

```
jacquesmenu@macstudio-1:/Volumes/JMI_Volume/JMI_Developpement/musicformats-git-dev/
    devtools > ls -sal
total 120
 0 drwxr-xr-x  12 jacquesmenu  staff    384 Feb 15 10:53 .
 0 drwxr-xr-x  31 jacquesmenu  staff    992 Feb 28 15:52 ..
 8 -rwxr-xr-x@  1 jacquesmenu  staff   1568 Jan 29 13:33 BuildMusicFormats.zsh
 8 -rwxr-xr-x@  1 jacquesmenu  staff   1662 Jan 29 13:33
    CreateMusicFormatsDocumentationPDFs.zsh
16 -rwxr-xr-x@  1 jacquesmenu  staff   4140 Jan 31 10:22 CreateMusicFormatsReleases.zsh
 8 -rwxr--r--@  1 jacquesmenu  staff    883 Feb 15 11:21 InstallLilyJazz.zsh
 8 -rwxr-xr-x@  1 jacquesmenu  staff   1231 Feb  6 13:58
    LocateFilesContainingFirstAndNotSecond.zsh
 8 -rwxr-xr-x   1 jacquesmenu  staff   1251 Jan 29 13:33 SetMusicFormatsVersionDate.zsh
 8 -rwxr-xr-x   1 jacquesmenu  staff   1304 Jan 29 13:33 SetMusicFormatsVersionNumber.zsh
 8 -rwxr-xr-x   1 jacquesmenu  staff    436 Jan 29 13:33 ShowMusicFormatsVersion.zsh
48 -rwxr-xr-x@  1 jacquesmenu  staff  22752 Feb 23 11:43 ZshDefinitionsForMusicFormats.zsh
```

14.1 Building the library

devtools/BuildMusicFormats.zsh executes `cmake`, whose configuration is in build/:

```
jacquesmenu@macstudio-1:/Volumes/JMI_Volume/JMI_Developpement/musicformats-git-dev/build >
     ls -sal
total 992
  0 drwxr-xr-x  13 jacquesmenu  staff     416 Feb 22 15:55 .
  0 drwxr-xr-x  31 jacquesmenu  staff     992 Feb 28 15:52 ..
 16 -rw-r--r--@  1 jacquesmenu  staff    6148 Feb 16 18:17 .DS_Store
904 -rw-r--r--@  1 jacquesmenu  staff  405409 Mar  1 06:03 BuildMusicFormats.zsh.log
  8 -rw-r--r--   1 jacquesmenu  staff    1815 Jan 29 13:33 Building.md
 40 -rw-r--r--@  1 jacquesmenu  staff   18499 Feb  6 12:48 CMakeLists.txt
  8 -rw-r--r--   1 jacquesmenu  staff     292 Jan 29 13:33 MakePkg.bat
 16 -rw-r--r--   1 jacquesmenu  staff    6227 Jan 29 13:33 Makefile
  0 drwxr-xr-x   3 jacquesmenu  staff      96 Jan 29 13:33 android
  0 drwxr-xr-x  27 jacquesmenu  staff     864 Feb 28 15:49 bin
  0 drwxr-xr-x   3 jacquesmenu  staff      96 Jan 29 13:33 docker
  0 drwxr-xr-x   4 jacquesmenu  staff     128 Feb 28 15:49 lib
  0 drwxr-xr-x   8 jacquesmenu  staff     256 Feb 22 15:55 libdir
```

14.2 Building the documenation

devtools/CreateMusicFormatsDocumentationPDFs.zsh executes lives up to its name. The LaTeX source files are in doc/:

```
jacquesmenu@macstudio-1:/Volumes/JMI_Volume/JMI_Developpement/musicformats-git-dev/
    documentation > ll
total 24
 0 drwxr-xr-x   12 jacquesmenu  staff    384 Jan 29 13:33:19 2023 ./
 0 drwxr-xr-x   31 jacquesmenu  staff    992 Feb 28 15:52:03 2023 ../
24 -rw-r--r--@   1 jacquesmenu  staff  10244 Feb 23 11:57:51 2023 .DS_Store
 0 drwxr-xr-x   22 jacquesmenu  staff    704 Jan 29 13:33:14 2023 CommonLaTeXFiles/
 0 drwxr-xr-x   32 jacquesmenu  staff   1024 Jan 29 13:33:14 2023 IntroductionToMusicXML/
 0 drwxr-xr-x   47 jacquesmenu  staff   1504 Feb 17 11:12:27 2023 mfapiguide/
 0 drwxr-xr-x    8 jacquesmenu  staff    256 Feb 17 11:12:49 2023 MusicFormatsFigures/
 0 drwxr-xr-x  105 jacquesmenu  staff   3360 Mar  1 06:12:41 2023 mfmaintainanceguide/
 0 drwxr-xr-x   54 jacquesmenu  staff   1728 Feb 17 11:13:30 2023 mfuserguide/
 0 drwxr-xr-x    3 jacquesmenu  staff     96 Jan 29 13:33:14 2023 builddoc/
 0 drwxr-xr-x   45 jacquesmenu  staff   1440 Jan 29 13:33:14 2023 mfgraphics/mfgraphics
 0 drwxr-xr-x    5 jacquesmenu  staff    160 Jan 29 13:33:14 2023 libmusicxml2Presentation
    /
```

When modifying a documentation's structure information, i.e. `\chapter`, `\section` and `\subsection` commands contents, `pdflatex` should be run three times to get the corresponding PDF.

14.3 Releases creation

devtools/CreateMusicFormatsReleases.zsh lives up to its name too.

14.4 Checking source files contents

devtools/LocateFilesContainingFirstAndNotSecond.zsh helps to find superfluous C++17 includes or other such checking tasks.

14.5 Handly ZSH functions and aliases

There are a couple of them in devtools/ZshDefinitionsForMusicFormats.zsh

Part IV

The two-phase visitors pattern

Chapter 15

The two-phase visitors pattern

MusicFormats uses a two-phase visitors pattern designed by Dominique Fober to traverse data structures such an `xmlElement tree` or an MSR description, handling each node in the structure in a systematic way. This is in contrast to a programmed top-down traversal.

Such data structures traversals is actually *data driven*: a visitor can decide to 'see' only selected node types.

There are case where visiting is not the way to go, see the sections below.

15.1 Basic mechanism

Visiting a node in a data structure is done in this order:

- first phase: visit the node for the fist time, top-down;
- visit the node contents, using the same two-phase visitors pattern;
- second phase: visit the node for the second time, bottom-up.

The first can be used to prepare data needed for the node contents visit, for example. Then the second phase can used such data, if relevant, as well as data created by the node contents visit, do consolidate the whole.

A visitor class should:

- inherit from `basevisitor`;
- inherit from the smart pointer classes it visits;
- define methods `visitStart ()` and/or `visitEnd ()` depending on which phases it wants to handle. The parameter of all such `visit* ()` methods is always a reference to a smart pointer.

`basevisitor` is defined in `libmusicxml/src/visitors!basevisitor.h`, and contains nothing:

```
class   basevisitor
{
  public:
    virtual ~basevisitor() {}
};
```

It is used as the base class of all visitors in `browsedata ()` methods:

```
void msrWords::acceptIn (basevisitor* v)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "% ==> msrWords::acceptIn ()";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  if (visitor<S_msrWords>*
    p =
      dynamic_cast<visitor<S_msrWords>*> (v)) {
        S_msrWords elem = this;

#ifdef MF_TRACE_IS_ENABLED
        if (gMsrOahGroup->getTraceMsrVisitors ()) {
          std::stringstream ss;

          ss <<
            "% ==> Launching msrWords::visitStart ()";

          gWaeHandler->waeTraceWithoutInputLocation (
            __FILE__, __LINE__,
            ss.str ());
        }
#endif // MF_TRACE_IS_ENABLED
        p->visitStart (elem);
  }
}
```

15.2 Browser template classes

There are several such classes, all with the same specification as the one in libmusicxml/src/lib!tree_browser.h, named to allow easy seach for them in the code base.
For example, in src/formats/msr/msrElements.h, there is:

```
//______________________________________________________________________________
template <typename T> class   msrBrowser : public browser <T>
{
  public:

    msrBrowser (basevisitor* v) : fVisitor (v) {}

    virtual                 ~msrBrowser () {}

  public:

    virtual void set (basevisitor* v) { fVisitor = v; }

    virtual void browse (T& t) {
      enter (t);

      t.browseData (fVisitor);

      leave (t);
    }

  protected:

    basevisitor*  fVisitor;

    virtual void enter (T& t) { t.acceptIn  (fVisitor); }
    virtual void leave (T& t) { t.acceptOut (fVisitor); }
};
```

15.3 A first example: counting notes in MusicMXL data

In libmusicxml/samples/countnotes.cpp, counting the notes in MusicXML data needs only see `S_note` nodes. class `countnotes` thus inherits only from a visitor for this type of node, and all the other node types are simply ignored.

vVsitor method `countnotes::visitStart` only has to increment the notes count:

Listing 15.1: `countnotes.cpp`

```
class   countnotes :
  public visitor<S_note>
{
  public:
    int fCount;

    countnotes() : fCount (0) {}

    virtual ~countnotes () {}

    void visitStart ( S_note& elt )   { fCount++; }
};
```

15.4 A more complex example

Let's look at the `<scaling/>` MusicXML element:

```
    <scaling>
      <millimeters>7</millimeters>
      <tenths>40</tenths>
    </scaling>
```

It contains a `<millimeter/>` and a `<tenth/>` element. The latter two don't contain any other elements, so `visitStart ()` is enough for them.

There is nothing to do on the visit start upon `<scaling/>`, so there is no such method. On the visit end upon `<scaling/>`, though, the values grabbed from the `<millimeter/>` and `<tenth/>` elements are used to create the class `msrScaling` description.

Should a visit start method have been written, the execution order would have been:

```
  mxsr2msrTranslator::visitStart ( S_scaling& elt)
    mxsr2msrTranslator::visitStart ( S_millimeters& elt )
    mxsr2msrTranslator::visitStart ( S_tenths& elt )
  mxsr2msrTranslator::visitEnd ( S_scaling& elt)
```

or, depending on the order in which the subelements of `<scaling/>` are visited:

```
  mxsr2msrTranslator::visitStart ( S_scaling& elt)
    mxsr2msrTranslator::visitStart ( S_tenths& elt )
    mxsr2msrTranslator::visitStart ( S_millimeters& elt )
  mxsr2msrTranslator::visitEnd ( S_scaling& elt)
```

In src/passes/mxsr2msr/mxsr2msrTranslator.cpp, visiting a <scaling/> element is handled this way:

Listing 15.2: Visiting <scaling />

```
void mxsr2msrTranslator::visitStart ( S_millimeters& elt )
{
#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting S_millimeters" <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fCurrentMillimeters = (float)(*elt);
}

void mxsr2msrTranslator::visitStart ( S_tenths& elt )
{
#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting S_tenths" <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fCurrentTenths = (float)(*elt);
}

void mxsr2msrTranslator::visitEnd ( S_scaling& elt)
{
  int inputLineNumber =
    elt->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> End visiting S_scaling" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

```

```
  // create a scaling
  S_msrScaling
    scaling =
      msrScaling::create (
        inputLineNumber,
        fCurrentMillimeters,
        fCurrentTenths);

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceGeometry ()) {
    gLog <<
      "There are " << fCurrentTenths <<
      " tenths for " <<  fCurrentMillimeters;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // set the MSR score's scaling
  fMsrScore->
    setScaling (scaling);
}
```

15.5 Data browsing order

The order of the visit of a node's subnodes is programmed in `browseData ()` methods, such as:

Listing 15.3: `msrDoubleTremolo::browseData (basevisitor* v)`

```
void msrDoubleTremolo::browseData (basevisitor* v)
{
  if (fDoubleTremoloFirstElement) {
    // browse the first element
    msrBrowser<msrElement> browser (v);
    browser.browse (*fDoubleTremoloFirstElement);
  }

  if (fDoubleTremoloSecondElement) {
    // browse the second element
    msrBrowser<msrElement> browser (v);
    browser.browse (*fDoubleTremoloSecondElement);
  }
}
```

Since this order is set in the `browsedata ()` methods, it cannot be influenced by the visitors of the corresponding class instances.

There are cases where the data should be sorted prior to being browsed, such as the staves in parts: this ensures that they are browsed in this order: harmonies staff, other staves, figured bass staff.

15.6 Selectively inhibiting data browsing

15.6.1 Inhibiting data browsing in the code

In some cases, it is desirable not to browse part of the data. This is the case when a given class contains non-normalized data, i.e. data that occurs elsewhere and will be browsed in another class instance.

For example, class `msrMultiMeasureRest` contains class `msrMeasure` instances. class `msrScore` contains:

```
    // in <multiple-rest/>, the multi-measure rests are explicit,
    // whereas LilyPond only needs the number of multi-measure rests
    Bool                    fInhibitMultiMeasureRestsBrowsing;

    void                    setInhibitMultiMeasureRestsBrowsing ()
                                {
                                  fInhibitMultiMeasureRestsBrowsing = true;
                                }

    Bool                    getInhibitMultiMeasureRestsBrowsing () const
                                {
                                  return
                                    fInhibitMultiMeasureRestsBrowsing;
                                };
```

Class `lpsr2lilypondTranslator` checks this setting:

```
void lpsr2lilypondTranslator::visitEnd (S_msrNote& elt)
{
  // ... ... ...

  if (fOnGoingMultiMeasureRests) {
    switch (elt->getNoteKind ()) {
      case msrNoteKind::kNoteRestInMeasure:
        // don't handle multi-measure restss, that's done in visitEnd (
    S_msrMultiMeasureRest&)
          if (elt->getNoteOccupiesAFullMeasure ()) {
            Bool inhibitMultiMeasureRestsBrowsing =
              fVisitedLpsrScore->
                getMsrScore ()->
                  getInhibitMultiMeasureRestsBrowsing ();

            if (inhibitMultiMeasureRestsBrowsing) {
#ifdef MF_TRACE_IS_ENABLED
              if (
                gTraceOahGroup->getTraceNotes ()
                  ||
                gTraceOahGroup->getTraceMultiMeasureRests ()
              ) {
                gLog <<
                  "% ==> end visiting multi-measure rests is ignored" <<
                  std::endl;
              }
#endif // MF_TRACE_IS_ENABLED

#ifdef MF_TRACE_IS_ENABLED
              if (gTraceOahGroup->getTraceNotesDetails ()) {
                gLog <<
                  "% ==> returning from visitEnd (S_msrNote&)" <<
                  std::endl;
              }
#endif // MF_TRACE_IS_ENABLED

              noteIsToBeIgnored = true;
            }
          }
        break;
  // ... ... ...
}
```

Another example is in the class `lpsr2lilypondTranslator` constructor:

```
lpsr2lilypondTranslator::lpsr2lilypondTranslator (
  S_lpsrScore      lpsrScore,
  const S_msrOahGroup&  msrOpts,
  const S_lpsrOahGroup& lpsrOpts,
  std::ostream&        lilypondCodeStream)
  : fLilypondCodeStream (
      lilypondCodeStream)
{
  fMsrOahGroup  = msrOpts;
  fLpsrOahGroup = lpsrOpts;

  // the LPSR score we're visiting
  fVisitedLpsrScore = lpsrScore;

  // inhibit the browsing of grace notes groups before,
  // since they are handled at the note level
  fVisitedLpsrScore->
    getMsrScore ()->
      setInhibitGraceNotesGroupsBeforeBrowsing ();

  // inhibit the browsing of grace notes groups after,
  // since they are handled at the note level
  fVisitedLpsrScore->
    getMsrScore ()->
      setInhibitGraceNotesGroupsAfterBrowsing ();
```

The test for browsing inhibition is done in src/formats/msr/msrNotes.cpp:

```
void msrNote::browseData (basevisitor* v)
{
  // browse the grace notes group before note if any
  if (fGraceNotesGroupBeforeNote) {
    // fetch the score
    S_msrScore
      score =
        fetchUpLinkToNoteToScore ();

    if (score) {
      Bool
        inhibitGraceNotesGroupsBeforeBrowsing =
          score->
            getInhibitGraceNotesGroupsBeforeBrowsing ();

      if (inhibitGraceNotesGroupsBeforeBrowsing) {
#ifdef MF_TRACE_IS_ENABLED
        if (
          gMsrOahGroup->getTraceMsrVisitors ()
            ||
          gTraceOahGroup->getTraceNotes ()
            ||
          gTraceOahGroup->getTraceGraceNotes ()
        ) {
          std::stringstream ss;

          ss <<
            "% ==> visiting grace notes groups before notes is inhibited";

          gWaeHandler->waeTraceWithoutInputLocation (
            __FILE__, __LINE__,
            ss.str ());
        }
#endif // MF_TRACE_IS_ENABLED
      }
      else {
        // browse the grace notes group before note
        msrBrowser<msrGraceNotesGroup> browser (v);
        browser.browse (*fGraceNotesGroupBeforeNote);
      }
    }
  }

  // ... ... ...
}
```

15.6.2 Inhibiting data browsing by options

Choosing which elements to browse can be more selective:

```
void msrStaff::browseData (basevisitor* v)
{
  // ... ... ...

  if (fStaffAllVoicesList.size ()) {
    for (const S_msrVoice& voice : fStaffAllVoicesList) {
      // is this voice name in the ignore voices set?
      Bool ignoreVoice (false);

      std::string voiceName =
        voice->
          getVoiceName ();

      const std::set<std::string>&
        ignoreMsrVoicesSet =
          gGlobalMsr2msrOahGroup->
            getIgnoreMsrVoicesSet ();

      // ... ... ...

      if (ignoreMsrVoicesSet.size ()) {
        ignoreVoice =
          mfStringIsInStringSet (
            voiceName,
            ignoreMsrVoicesSet);
      }

      if (ignoreVoice) {
#ifdef MF_TRACE_IS_ENABLED // JMI
        if (gTraceOahGroup->getTraceVoices ()) {
          std::stringstream ss;

          ss <<
            "Ignoring voice \"" <<
            voiceName <<
            "\"";

          gWaeHandler->waeTraceWithoutInputLocation (
            __FILE__, __LINE__,
            ss.str ());
        }
#endif // MF_TRACE_IS_ENABLED
      }

      else {
        msrBrowser<msrVoice> browser (v);
        browser.browse (*voice);
      }
    } // for
  }

  // ... ... ...
}
```

15.7 Adapting visitors to data browsing order with booleans

A frequent situation is when the visitor's actions should be tuned depending upon which elements are being visited. In simple case, this can be handled with boolean variables.

For example, `<system-layout/>` may occur both in the `<defaults/>` and `<print/>` MusicXML markups:

```
<defaults>
  <scaling>
    <millimeters>7.3</millimeters>
    <tenths>40</tenths>
  </scaling>
  <page-layout>
    <page-height>1534</page-height>
    <page-width>1151</page-width>
    <page-margins type="both">
      <left-margin>54.7945</left-margin>
      <right-margin>54.7945</right-margin>
      <top-margin>27.3973</top-margin>
      <bottom-margin>27.3973</bottom-margin>
    </page-margins>
  </page-layout>
  <system-layout>
    <system-margins>
      <left-margin>15</left-margin>
      <right-margin>0</right-margin>
    </system-margins>
    <system-distance>92.5</system-distance>
    <top-system-distance>27.5</top-system-distance>
  </system-layout>

// ... ... ...

<part id="P1">
  <measure number="1">
    <print>
      <system-layout>
        <system-margins>
          <left-margin>75.625</left-margin>
          <right-margin>0</right-margin>
        </system-margins>
        <top-system-distance>410.9375</top-system-distance>
      </system-layout>
      <staff-layout>
        <?DoletSibelius JustifyAllStaves=false?>
        <?DoletSibelius ExtraSpacesAbove=3?>
      </staff-layout>
      <measure-layout>
        <measure-distance>20</measure-distance>
      </measure-layout>
    </print>
```

To know which element is being visited, we use boolean `fOnGoing*` variables, such as `fOnGoingPrintLayout` in class `msr2mxsrTranslator`.

It is assigned in:

```
void msr2mxsrTranslator::visitStart (S_msrPrintLayout& elt)
{
 // ... ... ...

 fOnGoingPrintLayout = true;
}

void msr2mxsrTranslator::visitEnd (S_msrPrintLayout& elt)
```

```
{
  // ... ... ...

  fOnGoingPrintLayout = false;
}
```

and checked for example in:

```
void msr2mxsrTranslator::visitStart (S_msrSystemLayout& elt)
{
  // ... ... ...

  // create a system layout element
  Sxmlelement
    systemLayoutElement =
      createMxmlElement (k_system_layout, "");

  if (fOnGoingPrintLayout) {
    // append it to the current print element
    fCurrentPrintElement->push (
      systemLayoutElement);
  }
  else {
    // don't append it at once to the score defaults element
    fScoreDefaultsSystemLayoutElement = systemLayoutElement;
  }
```

When the data browsing order does not fit the needs of a visitor, the latter has to store the values gathered until they can be processed. This occurs for exemple in `mxsr2msrTranslator`, which uses `fCurrentPrintLayout` for this purpose:

```
void mxsr2msrTranslator::visitStart ( S_system_layout& elt )
{
  // ... ... ...

  // create the system layout
  fCurrentSystemLayout =
    msrSystemLayout::create (
      inputLineNumber);

  fOnGoingSystemLayout = true;
}

void mxsr2msrTranslator::visitEnd ( S_system_layout& elt )
{
  // ... ... ...

  if (fOnGoingPrint) {
    // set the current print layout's system layout
    fCurrentPrintLayout->
      setSystemLayout (
        fCurrentSystemLayout);
  }
  else {
    // set the MSR score system layout
    fMsrScore->
      setSystemLayout (
        fCurrentSystemLayout);
  }

  // forget about the current system layout
  fCurrentSystemLayout = nullptr;

  fOnGoingSystemLayout = false;
}
```

15.8 Adapting visitors to data browsing order with stacks

In more complex cases, the visiting order leads to have several on-going elements simultaneously. This is the case with class `msrTuplet`, which can be nested.

They are handled in `src/passes/mxsr2msr/mxsr2msrTranslator` and `src/passes/lpsr2lilypond/lpsr2lilypond` for example, using a stack to keep track of them.

MusicFormats never uses C++ STL stacks, because they cannot be iterated over:

```
    std::list<S_msrTuplet>    fOnGoingTupletsStack;
```

```
void lpsr2lilypondTranslator::visitStart (S_msrTuplet& elt)
{
  // ... ... ...

  if (fOnGoingTupletsStack.size ()) {
    // elt is a nested tuplet

    S_msrTuplet
      containingTuplet =
        fOnGoingTupletsStack.front ();

    // unapply containing tuplet factor,
    // i.e 3/2 inside 5/4 becomes 15/8 in MusicXML...
    elt->
      unapplySoundingFactorToTupletMembers (
        containingTuplet->
          getTupletFactor ());
  }

  // ... ... ...

  // push the tuplet on the tuplets stack
  fOnGoingTupletsStack (elt);

  // ... ... ...
}

void lpsr2lilypondTranslator::visitEnd (S_msrTuplet& elt)
{
  // ... ... ...

  // pop the tuplet from the tuplets stack
  fOnGoingTupletsStack ();

  // ... ... ...
}
```

15.9 Avoiding the visiting pattern by cascading

There are cases where we need a deterministic traversal of some data handled by MusicFormats. For example, appending a `msrStaffDetails` instance to a part should be cascaded to its staves. It would be an overkill to create a specific browser for this purpose.

This is what method `msrPart::appendStaffDetailsToPart ()` does:

```
void msrPart::appendStaffDetailsToPart (
  const S_msrStaffDetails& staffDetails)
{
  // ... ... ...

  // register staff details in part
  fCurrentPartStaffDetails = staffDetails;

  // append staff details to registered staves
  for (
    std::map<int, S_msrStaff>::const_iterator i =
      getPartStaveNumbersToStavesMap.begin ();
    i != getPartStaveNumbersToStavesMap.end ();
    ++i
  ) {
    S_msrStaff
      staff = (*i).second;

    staff->
      appendStaffDetailsToStaff (
        staffDetails);
  } // for
}
```

Another case is the handling the various elements attached to an class `msrNote` instance, among them chords, grace notes groups and tuplet, all of which contain notes too.
Doing things in the right order can be tricky, see `src/passes/lpsr2lilypond/lpsr2lilypondTranslator.h/.cpp`.

The time-oriented representation of scores in MSR is also printed by cascadingthrough `printSlices ()` methods, see chapter 21 [MSR time-oriented represention], page 218.

Part V

MusicFormats components

Chapter 16

MusicFormats components (MFC)

MusicFormats supports keeping the history of its components versions using a dedicated representation, as an alternative to separate release notes. The source files are in src/components/.

16.1 Versions numbers

The basic data structure is class `mfcVersionNumber`:

```
class   mfcVersionNumber : public smartable
{
  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    Bool                  operator== (const mfcVersionNumber& other) const;

    Bool                  operator!= (const mfcVersionNumber& other) const;

    Bool                  operator<  (const mfcVersionNumber& other) const;

    Bool                  operator>= (const mfcVersionNumber& other) const;

    Bool                  operator>  (const mfcVersionNumber& other) const;

    Bool                  operator<= (const mfcVersionNumber& other) const;

  public:

    // print
    // ------------------------------------------------------

    std::string           asString () const;

    void                  print (std::ostream& os) const;

  private:

    // fields
    // ------------------------------------------------------

    int                   fMajorNumber;
    int                   fMinorNumber;
    int                   fPatchNumber;
    std::string           fPreRelease;
};
```

16.2 Versions descriptions

Each component version is described by a class `mfcVersionDescr` instance:

```
class   mfcVersionDescr : public smartable
{
  // ... ... ...

  private:

    // fields
    // ------------------------------------------------------

    S_mfcVersionNumber    fVersionNumber;
    std::string           fVersionDate;
    std::list<std::string>
                          fVersionDescriptionItems;
};
```

16.3 Versions histories

An instance of `mfcVersionsHistory` is essentially a list of `mfcVersionsHistory` instances:

```
class   mfcVersionsHistory : public smartable
{
  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    void                  appendVersionDescrToHistory (
                            const S_mfcVersionDescr& versionDescr);

    S_mfcVersionDescr     fetchMostRecentVersion () const;

    S_mfcVersionNumber    fetchMostRecentVersionNumber () const;

  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    std::list<S_mfcVersionDescr>
                          fVersionsList;
};
```

The current version of a component is the last one appended to `fVersionsList`:

```
S_mfcVersionDescr mfcVersionsHistory::fetchMostRecentVersion () const
{
#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  mfAssert (
    __FILE__, __LINE__,
    fVersionsList.size () > 0,
    "fVersionsList is empty");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  return fVersionsList.back ();
}
```

16.4 Components descriptions

The components of MusicFormats are described by enumeration type `mfcComponentKind`:

```
enum class mfcComponentKind {
  kComponentRepresentation,
  kComponentPass,
  kComponentGenerator,
  kComponentConverter,
  kComponentLibrary
};
```

The purely virtual class `mfcComponent` is a superclass to the ones describing formats, passes, generators, converters and the MusicFormats library itself:

```
class   mfcComponent : public smartable
{
  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    S_mfcVersionDescr     fetchComponentMostRecentVersion () const
                              {
                                return
                                  fVersionsHistory->
                                     fetchMostRecentVersion ();
                              }

  // ... ... ...

  public:

    // print
    // ------------------------------------------------------

    std::string           asString () const;

    std::string           mostRecentVersionNumberAndDateAsString () const;

    virtual void          print (std::ostream& os) const;

    virtual void          printVersion (std::ostream& os) const;
    virtual void          printHistory (std::ostream& os) const;

  protected:

    // protected services
    // ------------------------------------------------------

    virtual void          printOwnHistory (std::ostream& os) const;

  protected:

    // protected fields
    // ------------------------------------------------------

    std::string           fComponentName;

    mfcComponentKind       fComponentKind;

    S_mfcVersionsHistory  fVersionsHistory;
};
```

The virtual `printVersion ()` and `printHistory ()` methods are called by the `--v, --version` and `--hist, --history` options to the various generators and converters.

Representations and passes have a single, linear history, whereas the generators, the converters and MusicFormats itself use several of them, each with its own history. This leads to a hierarchy of classes:

- class `mfcRepresentationComponent` for formats;
- class `mfcPassComponent` for passes;
- purely virtual class `mfcMultiComponent` for the generators, converters and MusicFormats library, itself the superclass of:
  - class `mfcGeneratorComponent`;
  - class `mfcConverterComponent`;
  - class `mfcLibraryComponent`.

Multi-components have their own history, hence field method `mfcComponent::printOwnHistory ()`. Class `mfcMultiComponent` is described below.

16.5 Multi-components

Class `mfcMultiComponent` contains lists of the formats and passes used:

```
class   mfcMultiComponent : public mfcComponent
{
  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    std::list<S_mfcRepresentationComponent>
                          fRepresentationComponentsList;
    std::list<S_mfcPassComponent>
                          fPassComponentsList;

    // should the version number be at least equal to
    // the ones of the components?
    mfcMultiComponentEntropicityKind
                          fComponentEntropicityKind;

    mfcMultiComponentUsedFromTheCLIKind
                              fComponentUsedFromTheCLIKind;
};
```

Enumeration type `mfcMultiComponentEntropicityKind` is used to check that the version number of a `mfcMultiComponent` instance is at least equal to the version numbers of the formats and passes it uses:

```
enum class mfcMultiComponentEntropicityKind {
  kComponentEntropicityYes,
  kComponentEntropicityNo
};
```

Enumeration type `mfcMultiComponentUsedFromTheCLIKind` is used to display context sensitive output with the `-version, -v` and `-history, -hist` options when the library is used from command line services or through the functional API:

```
enum class mfcMultiComponentUsedFromTheCLIKind {
  kComponentUsedFromTheCLIYes,
  kComponentUsedFromTheCLINo
};
```

This allows for the maintainers of little used tools not to worry about using components with version numbers greater than their own.
Only constant `mfcMultiComponentUsedFromTheCLIKind::kComponentUsedFromTheCLIYes` is used at the time of this writing.

Method `mfcMultiComponent::print ()` displays the regular version numbers:

```
jacquesmenu@macmini: ~ > xml2ly -version
Command line version of musicxml2lilypond converter v0.9.51 (October 12 2021)

Representations versions:
  MXSR
    v0.9.5 (October 6 2021)
  MSR
    v0.9.51 (October 14 2021)
  LPSR
    v0.9.5 (October 6 2021)

Passes versions:
  mxsr2msr
    v0.9.5 (October 6 2021)
  msr2msr
    v0.9.5 (October 6 2021)
  msr2lpsr
    v0.9.5 (October 6 2021)
  lpsr2lilypond
    v0.9.5 (October 6 2021)
```

Method `mfcMultiComponent::printHistory ()` displays information analogous to release notes:

```
jacquesmenu@macmini: ~ > xml2brl -history
Command line version of musicxml2braille converter v0.9.51 (October 12 2021)

Own history:
  v0.9.5 (October 6 2021):
    Start of sequential versions numbering

  v0.9.51 (October 12 2021):
    Fixed trace OAH issue

Representations history:
  MXSR
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  MSR
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

    v0.9.51 (October 14 2021):
      Refined MSR names and summary display options

  BSR
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

Passes history:
  mxsr2msr
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  msr2msr
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  msr2bsr
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  bsr2bsr
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  bsr2braille
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering
```

16.6 Versions history creation

MusicFormats uses *semantic* version numbering, such as `v0.9.61`:

- the library itself gets a new number right after a new branch as been created for it. Branching to ”v0.9.61” causes the library to be numbered ”v0.9.61” with `SetMusicFormatsVersionNumber.bash`;
- each representation, converter or pass that is modified this new branch has been created gets a new history element with the same number as the library.

The versions history must exist before the `-version, -v` and `-history, -hist` options are handled. They are thus created early by specific functions, placed in `*Component.h/.cpp` files.

The functions that create them ensure than that is done at most once.

16.6.1 Representations and passes components creation

This is done in `create*RepresentationComponent ()` and `create*PassComponent ()` functions, respectively.

For example, MSR versions are handled by function `createMsrRepresentationComponent ()` in `src/formats/msr/msrHistory.h/.cpp`:

```
S_mfcRepresentationComponent EXP createMsrRepresentationComponent ()
{
  static S_mfcRepresentationComponent pRepresentationComponent;

  // protect library against multiple initializations
  if (! pRepresentationComponent) {

#ifdef MF_TRACE_IS_ENABLED
    if (gEarlyOptions.getEarlyTraceComponents ()) {
      std::stringstream ss;

      ss <<
        "Initializing MSR format component";

      gWaeHandler->waeTraceWithoutInputLocation (
        __FILE__, __LINE__,
        ss.str ());
    }
#endif // MF_TRACE_IS_ENABLED

    // create the format component
    pRepresentationComponent =
      mfcRepresentationComponent::create (
        "MSR");

    // populate it
    pRepresentationComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.50"),
          "October 6, 2021",
          std::list<std::string> {
            "Start of sequential versions numbering"
          }
      ));

    pRepresentationComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.51"), // JMI
          "October 14, 2021",
          std::list<std::string> {
            "Refined MSR names and summary display options"
          }
      ));
  }

  return pRepresentationComponent;
}
```

The conversion of MusicXML to MXSRdoes not belong to MusicFormats since it is provided by `libmusicxml2`.

16.6.2 Generators and converters components creation

In that case, the formats and passes components used by the multi-component should be created as well.

For example, the formats and passes used by the `musicxml2braille` converter are appended to the atoms versions list in its history in function `createMusicxml2brailleConverterComponent ()` in `src/converters/musicxml2braille/musicxml2brailleConverterComponent.cpp`:

```
S_mfcConverterComponent EXP createMusicxml2brailleConverterComponent ()
{
  static S_mfcConverterComponent pConverterComponent;

  // protect library against multiple initializations
  if (! pConverterComponent) {

#ifdef MF_TRACE_IS_ENABLED
    if (gEarlyOptions.getEarlyTraceComponents ()) {
      std::stringstream ss;

      ss <<
        "Creating the musicxml2braille component";

      gWaeHandler->waeTraceWithoutInputLocation (
        __FILE__, __LINE__,
        ss.str ());
    }
#endif // MF_TRACE_IS_ENABLED

    // create the converter component
    pConverterComponent =
      mfcConverterComponent::create (
        "musicxml2braille",
        mfcMultiComponentEntropicityKind::kComponentEntropicityNo,
        mfcMultiComponentUsedFromTheCLIKind::kComponentUsedFromTheCLIYes); // JMI ???

    // populate the converter's own history
    pConverterComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.50"),
          "October 6, 2021",
          std::list<std::string> {
            "Start of sequential versions numbering"
          }
      ));

    pConverterComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.51"),
          "October 12, 2021",
          std::list<std::string> {
            "Fixed trace OAH issue"
          }
      ));

    // populate the converter's formats list
    pConverterComponent->
      appendRepresentationToMultiComponent (
        createMxsrRepresentationComponent ());
    pConverterComponent->
      appendRepresentationToMultiComponent (
        createMsrRepresentationComponent ());
    pConverterComponent->
      appendRepresentationToMultiComponent (
        createBsrRepresentationComponent ());

    pConverterComponent->
      appendPassToMultiComponent (
        createMxsr2msrComponent ());

    pConverterComponent->
      appendPassToMultiComponent (
        createMsr2msrComponent ());

    pConverterComponent->
      appendPassToMultiComponent (
        createMsr2bsrComponent ());

    pConverterComponent->
      appendPassToMultiComponent (
        createBsr2bsrComponent ());

    pConverterComponent->
      appendPassToMultiComponent (
        createBsr2brailleComponent ());
  }

  return pConverterComponent;
}
```

16.6.3 MusicFormats library component creation

This is done in function `createLibraryComponent ()` in `src/utilities/mfcLibraryComponent.h/.cpp`:

```
S_mfcLibraryComponent EXP createLibraryComponent ()
{
  static S_mfcLibraryComponent pLibraryComponent;

  // protect library against multiple initializations
  if (! pLibraryComponent) {

#ifdef MF_TRACE_IS_ENABLED
    if (gEarlyOptions.getTraceEarlyOptions ()) {
      std::stringstream ss;

      ss <<
        "Creating the  MFC library component";

      gWaeHandler->waeTraceWithoutInputLocation (
        __FILE__, __LINE__,
        ss.str ());
    }
#endif // MF_TRACE_IS_ENABLED

    // create the library's history
    pLibraryComponent =
      mfcLibraryComponent::create (
        "musicformats",
        mfcMultiComponentEntropicityKind::kComponentEntropicityNo,
        mfcMultiComponentUsedFromTheCLIKind::kComponentUsedFromTheCLIYes); // JMI ???

    // populate the library's own history
    pLibraryComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.50"),
          "October 6, 2021",
          std::list<std::string> {
            "Start of sequential versions numbering"
          }
      ));

    pLibraryComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.51"),
          "October 12, 2021",
          std::list<std::string> {
            "Adding a version number to the MusicFormats library",
            "Fixed trace OAH issue in the musicxml2* converters)"
          }
      ));

    pLibraryComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.52"),
          "October 12, 2021",
          std::list<std::string> {
            "Added MusicFormats library versions history to '-hist, -history'"
          }
      ));

    pLibraryComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.53"),
          "October 22, 2021",
          std::list<std::string> {
            "Replaced bool by class   Bool in variables and fields",
            "Created MFC (MusicFormats components)"
          }
      ));

    pLibraryComponent->
      appendVersionDescrToComponent (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.54"),
          "Novermber 6, 2021",
          std::list<std::string> {
            "Replaced std::cout and std::cerr by gOutput and gLog respectively in the CLI
    samples",
            "Finalized components numbering (MFC)"
          }
      ));

    // populate the library's components history
    pLibraryComponent->
      appendRepresentationToMultiComponent (
        createMsrRepresentationComponent ());
    pLibraryComponent->
      appendRepresentationToMultiComponent (
        createLpsrRepresentationComponent ());
    pLibraryComponent->
      appendRepresentationToMultiComponent (
        createBsrRepresentationComponent ());
    pLibraryComponent->
      appendRepresentationToMultiComponent (
        createMxsrRepresentationComponent ());

    pLibraryComponent->
      appendPassToMultiComponent (
        createMsr2msrComponent ());

    pLibraryComponent->
      appendPassToMultiComponent (
        createMsr2lpsrComponent ());
    pLibraryComponent->
      appendPassToMultiComponent (
        createLpsr2lilypondComponent ());

    pLibraryComponent->
      appendPassToMultiComponent (
        createMsr2bsrComponent ());
    pLibraryComponent ->
      appendPassToMultiComponent (
        createBsr2bsrComponent ());
    pLibraryComponent ->
      appendPassToMultiComponent (
        createBsr2brailleComponent ());

    pLibraryComponent ->
      appendPassToMultiComponent (
        createMsr2mxsrComponent ());

    pLibraryComponent ->
      appendPassToMultiComponent (
        createMxsr2musicxmlComponent ());

    pLibraryComponent ->
      appendPassToMultiComponent (
        createMxsr2guidoComponent ());
  }

  return pLibraryComponent;
}
```

Functions `createLibraryComponent ()` is called in `src/clisamples/displayMusicformatsVersion.cpp` and `src/clisamples/displayMusicformatsHistory.cpp`.

16.6.4 Version and history options handling

In order to be able to execute the `-version`, `-v` and `-history`, `-hist` options of a generator or converter, a `oahHandler` instance must be supplied with a `mfcMultiComponent` instance.

Field `oahHandler::fHandlerMultiComponent` is used for this purpose:

```
//______________________________________________________________________________
class EXP oahHandler : public smartable
{
  // ... ... ...

  protected:

    // protected initialization
    // ------------------------------------------------------

    virtual void          initializeHandlerMultiComponent () = 0;

  public:

    // set and get
    // ------------------------------------------------------

    // ... ... ...

    S_mfcMultiComponent   getHandlerMultiComponent () const
                              { return fHandlerMultiComponent; }

  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    // ... ... ...

    // compound versions
    S_mfcMultiComponent    fHandlerMultiComponent;
};
```

Field `oahHandler::fHandlerMultiComponent` is set in the `oahHandler` sub-classes constructors by a call to the overriden `initializeHandlerMultiComponent ()`.

For example in constructor `xml2xmlInsiderHandler::xml2xmlInsiderHandler ()`:

```
xml2xmlInsiderHandler::xml2xmlInsiderHandler (
  const std::string& serviceName,
  std::string handlerHeader)
  : oahInsiderHandler (
      serviceName,
      handlerHeader,
R"(
          Welcome to the MusicXML to MusicXML converter
          delivered as part of the MusicFormats library.

      --- https://github.com/jacques-menu/musicformats ---
)",
R"(
Usage: xml2xml [[option]* [MusicXMLFile|-] [[option]*
)")
{
  // ... ... ...

  // initialize the multi-component
  initializeHandlerMultiComponent ();

  // ... ... ...
}
```

The overriden `initializeHandlerMultiComponent ()` methods merely get the atom or compound versions to assign it to field `oahHandler::fHandlerMultiComponent`.

For example, for `Mikrokosmos3Wandering`, the compound versions is simply set in the corresponding insider class `Mikrokosmos3WanderingInsiderHandler`:

```
void Mikrokosmos3WanderingInsiderHandler::initializeHandlerMultiComponent ()
{
  fHandlerMultiComponent =
    createMikrokosmos3WanderingGeneratorComponent ();
}
```

16.7 Accessing versions in regular handlers

A regular handler merely gets the compound versions of the insider handler it relies upon in its overriden `initializeHandlerMultiComponent ()` method:

```
class EXP oahRegularHandler : public oahHandler
/*
  A regular OAH handler relies on the existence of so-called 'insider' handler,
  that contains all the options values gathered from the user,
  grouped according to the internal representations and passes used.

  The variables containing the values of the options selected by the user
  are actually held by the insider handler.
*/
{
  // ... ... ... ...

  protected:

    // protected initialization
    // ------------------------------------------------------

  // ... ... ... ...

    void                      initializeHandlerMultiComponent () override
                                  {
                                  fHandlerMultiComponent =
                                    fInsiderHandler->
                                      getHandlerMultiComponent ();
                                }

  // ... ... ... ...
};
```

16.8 Getting current version numbers

Apart from the version and history options, such current version numbers may be used in the output from generators and converters, depending on the options. A component description is the way to achieve that in the latter two cases.

16.8.1 Current version numbers in options

Option `-version, -v` displays the versions of generators and converters:

```
jacquesmenu@macmini: ~/musicformats-git-dev/musicxml > xml2xml -version
Command line version of musicxml2musicxml converter v0.9.51 (October 12 2021)

Representations versions:
  MXSR
    v0.9.5 (October 6 2021)
  MSR
    v0.9.51 (October 14 2021)

Passes versions:
  mxsr2msr
    v0.9.5 (October 6 2021)
  msr2msr
    v0.9.5 (October 6 2021)
  msr2mxsr
    v0.9.5 (October 6 2021)
  mxsr2musicxml
    v0.9.5 (October 6 2021)
```

Option `-history, -hist` display the versions history of generators and converters:

```
jacquesmenu@macmini: ~/musicformats-git-dev/musicxml > xml2gmn -history
Command line version of musicxml2guido converter v0.9.51 (October 12 2021)

Own history:
  v0.9.5 (October 6 2021):
    Start of sequential versions numbering

  v0.9.51 (October 12 2021):
    Fixed trace OAH issue

Representations history:
  MXSR
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  MSR
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

    v0.9.51 (October 14 2021):
      Refined MSR names and summary display options

Passes history:
  mxsr2msr
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  msr2msr
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  msr2mxsr
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering

  mxsr2guido
    v0.9.5 (October 6 2021):
      Start of sequential versions numbering
```

In src/oah/oahAtomsCollection.h/.cpp, class oahVersionAtom contains method printVersion ():

```
class EXP oahVersionAtom : public oahPureHelpValueLessAtom
{
  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    void                  applyValueLessAtom (std::ostream& os) override;

  // ... ... ...

  public:

    // print
    // ------------------------------------------------------

    // ... ... ...

    void                  printVersion (std::ostream& os) const;
};
```

The option is applied by method oahVersionAtom::applyElement ():

```
void oahVersionAtom::applyValueLessAtom (std::ostream& os)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gEarlyOptions.getTraceEarlyOptions ()) {
    std::stringstream ss;

    ss <<
      "==> option '" << fetchNames () << "' is a oahVersionAtom";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  int saveIndent = gIndenter.getIndent ();

  gIndenter.resetToZero ();

  printVersion (os);

  gIndenter.setIndent (saveIndent);

  fSelected = true;
}
```

The work is done by method `oahVersionAtom::printVersion ()`:

```
void oahVersionAtom::printVersion (std::ostream& os) const
{
  // get the handler version
  S_mfcMultiComponent
    handlerMultiComponent =
      fetchAtomUpLinkToHandler ()->
        getHandlerMultiComponent ();

#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  mfAssert (
    __FILE__, __LINE__,
    handlerMultiComponent != nullptr,
    "handlerMultiComponent is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  handlerMultiComponent->
    printVersion (os);
}
```

The situation is analog for histories with `printVersion ()` replaced by `printHistory ()`.

16.8.2 Current version numbers in formats

When creating LilyPond output, the current version number of the converter used is indicated as a comment when the option `-lilypond-generation-infos, -lpgi` option is used:

```
jacquesmenu@macmini: ~/musicformats-git-dev/musicxml > xml2ly --lilypond-generation-infos
    basic/HelloWorld.xml
\version "2.24.0"

% Pick your choice from the next two lines as needed
%myBreak = { \break }
myBreak = {}

% Pick your choice from the next two lines as needed
%myPageBreak = { \pageBreak }
myPageBreak = {}

% Generated by xml2ly v0.9.51 (October 12 2021)
% on Thursday 2021-11-11 @ 11:15:56 CET
% from "basic/HelloWorld.xml"

% ... ... ...
```

Class `lpsrScore` contains an MFC component field:

```
class EXP lpsrScore : public lpsrElement
{
  // ... ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // ... ... ... ...

    // the multi-component
    // ------------------------------------------------------
    S_mfcMultiComponent    fMultiComponent;

  // ... ... ... ...
};
```

In src/formats/lpsr//lpsrScores.cpp, constructor `lpsrScore::lpsrScore ()` stores the multi-component value and uses it to create an `lpsrComment` instance:

```
lpsrScore::lpsrScore (
  int                   inputLineNumber,
  const S_msrScore&     theMsrScore,
  const S_mfcMultiComponent& multiComponent)
    : lpsrElement (inputLineNumber)
{
  // ... ... ... ...

  fMsrScore = theMsrScore;

  fMultiComponent = multiComponent;

  // should the initial comments about the service and the options used
  // be generated?
  if (gGlobalLpsr2lilypondOahGroup->getXml2lyInfos ()) {
    // create the 'input source name and translation date' comment
    {
      std::stringstream ss;

      ss <<
        "Generated by " <<
        gOahOahGroup->getOahOahGroupServiceName () <<
        ' ' <<
        fMultiComponent->
          mostRecentVersionNumberAndDateAsString () <<
        std::endl <<

        "% on " <<
        gServiceRunData->getTranslationDateFull () <<
        std::endl <<

        "% from ";

      if (gServiceRunData->getInputSourceName () == "-") {
        ss << "standard input";
      }
      else {
        ss << "\"" << gServiceRunData->getInputSourceName () << "\"";
      }

      fInputSourceNameComment =
        lpsrComment::create (
          inputLineNumber,
          ss.str (),
          lpsrCommentGapAfterwardsKind::kCommentGapAfterwardsNo);
    }

    // ... ... ... ...
  }

  // ... ... ... ...
}
```

16.8.3 Current version numbers in passes

Another case is that of the generation of MusicXML output:

```
jacquesmenu@macmini: ~/musicformats-git-dev/musicxml > xml2xml -musicxml-generation-infos
    basic/HelloWorld.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 3.1 Partwise//EN"
      "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.1">
    <!--
==================================================
Created by xml2xml v0.9.5 (October 6 2021)
on Thursday 2021-11-11 @ 11:04:06 CET
from basic/HelloWorld.xml
==================================================
-->
    <work>
        <work-number/>
        <work-title>Hello World!</work-title>
    </work>
    <movement-number/>
    <movement-title/>
    <identification>
        <encoding>
            <software>xml2xml v0.9.5 (October 6 2021), https://github.com/jacques-menu/
    musicformats</software>
            <encoding-date>2021-11-10</encoding-date>
        </encoding>
        <miscellaneous>
            <miscellaneous-field name="description"/>
        </miscellaneous>
    </identification>

 <!-- ... ... ... -->
```

In src/passes/msr2mxsr/msr2mxsrTranslator.cpp, the start visitor of msrScore instances does that this way:

```
void msr2mxsrTranslator::visitStart (S_msrScore& elt)
{
  // ... ... ... ...

  // get the pass component
  S_mfcPassComponent
    passComponent =
      createMsr2mxsrComponent ();

  // get the pass component current version number and date
  std::string
    passComponentMostRecentVersionNumberAndDateAsString =
      passComponent->
        mostRecentVersionNumberAndDateAsString ();

  // create the initial creation comment
  std::stringstream ss;
  ss <<
    std::endl <<
    "==================================================" <<
    std::endl <<
    "Created by " <<
    gOahOahGroup->getOahOahGroupServiceName () <<
    ' ' <<
    passComponentMostRecentVersionNumberAndDateAsString <<
    std::endl <<

    "on " <<
    gServiceRunData->getTranslationDateFull () <<
    std::endl <<

    "from " <<
    gServiceRunData->getInputSourceName () <<
    std::endl <<

    "==================================================" <<
    std::endl;

  // append the initial creation comment to the score part wise element
  fResultingMusicxmlelement->push (createMxmlelement (kComment, ss.str ()));

  // create a software element
  Sxmlelement
    softwareElement =
      createMxmlelement (
        k_software,
        gOahOahGroup->getOahOahGroupServiceName ()
          + ' '
          + passComponentMostRecentVersionNumberAndDateAsString +
          ", https://github.com/jacques-menu/musicformats");

  // append it to the identification encoding
  appendToScoreIdentificationEncoding (softwareElement);

  // ... ... ... ...
}
```

Part VI

Options and help

Chapter 17

Options and help (OAH)

OAH is a powerful way of representating the options together with the corresponding help. The classical function `getopt* ()` family of functions are not up to the task because:

- there is a great number of options in MusicFormats;
- attaching the help to the options in a clean, neat way was highly desirable;
- more important still, the re-use of options whenever translators are combined into converters could only be achieved with an object oriented handling of the options and help.

The output of the help goes to standard output, so that the user can pipe it into a `more` or `less` command.

17.1 OAH basics

- OAH (Options And Help) is supposed to be pronounced something close to "whaaaah!" The intonation is left to the speaker, though... And as the saying goes: "OAH? why not!"
- options handling is organized as a hierarchical, introspective set of classes. An options and its corresponding help are grouped in a single object.
- the options can be supplied thru:
  - the command line, in `argv`. This allows for mixed options and arguments in any order, à la GNU;
  - the API functions such as function `musicxmlfile2lilypond ()`, in an options and arguments.
- class `oahElement` is the super-class of all options types, including groups and subgroups. It contains a short name and a long name, as well as a description. Short and long names can be used and mixed at will in the command line and in option vectors (API), as well as '`-`' and '`--`'. The short name is mandatory, but the long name may be empty if the short name is explicit enough.
- prefixes such '`-t=`' and `-help=`' allow for a contracted form of options. For example, `-t=meas,notes` is short for '`-t-meas, -tnotes`'. A `oahPrefix` contains the prefix name, the ersatz by which to replace it, and a description.
- a class `oahHandler` contains a list of `oahGroup`'s, each handled in a pair of `.h/.cpp` files such as `src/formats/msr/msrOah.h` and `src/formats/msr/msrOah.cpp`, and a list of options prefixes.
- a class `oahGroup` contains a list of `oahSubGroup`'s and an upLink to the containing `oahHandler`.
- a class `oahSubGroup` contains a list of `oahAtom`'s and an upLink to the containing `oahGroup`.
- each class `oahAtom` contains an atomic option and the corresponding help, and an upLink to the containing `oahSubGroup`.

17.2 Features

- the values of the various options can be displayed with the option `-display-options-values, -dov` option;
- partial help can be obtained, i.e. help about any group, subgroup or atom, showing the path in the hierarchy down to the corresponding option;
- there are various sub-classes of class `oahAtom` such as class `oahIntegerAtom`, class `oahBooleanAtom` and class `oahRationalAtom`, to control options values of common types;
- class `oahThreeBooleansAtom`, for example, allows for three boolean settings to be controlled at once with a single option;
- class `oahAtomStoringAValue` describes options for which a value is supplied in the command line or in option vectors (API);
- a class such as class `lpsrPitchesLanguageAtom` is used to supply a string value to be converted into an internal enumerated type;
- a class `oahCombinedBooleansAtom` contains a list of boolean atoms to manipulate several such atoms as a single one, see the 'class `cubase`' combined booleans atom in `src/passes/mxsr2msr/mxsr2msrOah.cpp`;
- class `oahMultiplexBooleansAtom` contains a list of boolean atoms sharing a common prefix to display such atoms in a compact manner, see the '`ignore-redundant-clefs`' multiplex booleans atom in `src/passes/mxsr2msr/mxsr2msrOah.cpp`;
- storing options and the corresponding help in class `oahGroup`'s makes it easy to re-use them. For example, file `xml2ly` and file `xml2lbr` have their three first passes in common, (up to obtaining the MSR description of the score), as well as the corresponding options and help;
- `src/oah/oahAtomsCollection` contains a bunch of general purpose options such as class `oahContactAtom`, class `oahFloatAtom` and class `oahLengthAtom`;
- a regular handler (used by default unless the option `-insider` is used), presents the options and help grouped by subject, such as voices and tuplets. It uses an insider handler, which groups them by internal representation and conversion pass. This is how options groups are re-used for various converters such as file `xml2ly`, file `xml2brl` and file `xml2xml`.

17.3 OAH classes inheritance

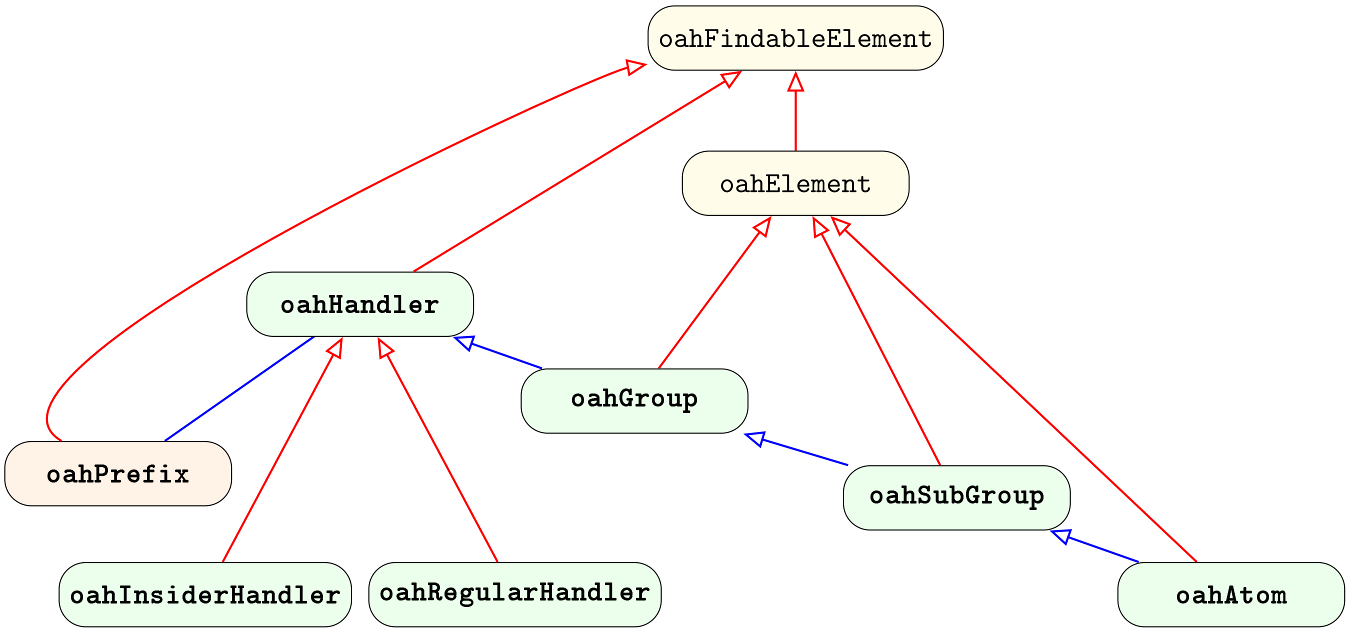
The picture at figure 17.1 [The OAH classes hierarchy], page 136, shows the hierarchy of the main OAH classes. The colors are used as follows:

The background colors are used as follows:

- green: a OAH element that is expected to be found in an options and help user view, such as class `oahHandler` and class `oahSubGroup`;
- pink: a complementary element provided by OAH, such as class `oahPrefix`;
- yellow: a base class with name class `oah*Element` for elements that can be used in another class, such as class `oahFindableElement`;

The arrows colors have the following meaning:

Figure 17.1: The OAH classes hierarchy



- red: a link from a class to its base class. For example, class `oahElement` is derived from class `oahFindableElement`;
- blue: a link from a class to another that uses smart pointers to one or more instances of the former. For example, an `msrTuplet` instance may be an element of an `msrGraceNotesGroup` instance.

There is a whole hierarchy of `oahAtom` sub-classes, some of which are provided in `src/oah/oahAtomsCollection.h.h/`
Here is the essentials of the OAH classes:

```
jacquesmenu@macmini:~/musicformats-git-dev/src/oah > gr Element | grep class
oahAtomsCollection.h:2085:class EXP oahNaturalNumbersSetElementAtom : public
    oahAtomStoringAValue
oahAtomsCollection.h:2271:class EXP oahIntSetElementAtom : public oahAtomStoringAValue
oahAtomsCollection.h:2357:class EXP oahStringSetElementAtom : public oahAtomStoringAValue
oahAtomsCollection.h:2450:class EXP oahStringToIntMapElementAtom : public
    oahAtomStoringAValue
oahAtomsCollection.h:2537:class EXP oahStringToStringMapElementAtom : public
    oahAtomStoringAValue
oahAtomsCollection.h:2624:class EXP oahStringToStringMultiMapElementAtom : public
    oahAtomStoringAValue
oahBasicTypes.h:373:class EXP oahPrefix : public oahFindableElement
oahBasicTypes.h:472:class EXP oahAtom : public oahElement
oahBasicTypes.h:994:class EXP oahSubGroup : public oahElement
oahBasicTypes.h:1163:class EXP oahGroup : public oahElement
oahBasicTypes.h:1396:class EXP oahHandler : public oahElement
oahElements.h:36:enum class oahElementValueKind {
oahElements.h:48:enum class oahElementVisibilityKind {
oahElements.h:60:enum class oahElementHelpOnlyKind {
oahElements.h:72:class   oahElement;
oahElements.h:82:class EXP oahFindableElement : public smartable
oahElements.h:206:class EXP oahElement : public oahFindableElement
oahElements.h:391:class EXP oahElementUse : public smartable
```

Class `oahFindableElement` is the base class for all those that can be introspected with the `-find` option.

17.4 Value-fitted atoms

Some options expect a value, such a length or a color, to be supplied as an argument in the command line or in a oahOptionsVector.

Purely virtual class oahValueFittedAtom, defined in src/oah/oahBasicTypes.h/.cpp, is a common ancestor to all the classes describing such options:

```
class EXP oahValueFittedAtom : public oahAtom
/*
  a purely virtual common ancestor for all atom classes
  that take a value from argv or an oahOptionsVector
*/
{
  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    void                      applyValueLessAtom (std::ostream& os) override;
                                // reports an error

    virtual void              applyAtomWithValue (
                                const std::string& theString,
                                std::ostream&      os) = 0;

    virtual void              applyAtomWithDefaultValue (std::ostream& os);
};
```

The classes derived from oahValueFittedAtom are:

```
jacquesmenu@macstudio:~/musicformats-git-dev/src > grh 'public oahValueFittedAtom'
oah/oahBasicTypes.h:class EXP oahAtomStoringAValue : public oahValueFittedAtom
oah/oahBasicTypes.h:class EXP oahPureHelpValueFittedAtom : public oahValueFittedAtom
oah/oahBasicTypes.h:class EXP oahAtomStoringAValue : public oahValueFittedAtom
oah/oahBasicTypes.h:class EXP oahPureHelpValueFittedAtom : public oahValueFittedAtom
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondFixedOctaveEntryAtom : public
    oahValueFittedAtom
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondFixedOctaveEntryAtom : public
    oahValueFittedAtom
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondFixedOctaveEntryAtom : public
    oahValueFittedAtom
```

For example:

```
class EXP lilypondFixedOctaveEntryAtom : public oahValueFittedAtom
{
  public:

    // creation
    // ------------------------------------------------------

    static SMARTP<lilypondFixedOctaveEntryAtom> create (
                                const std::string& longName,
                                const std::string& shortName,
                                const std::string& description,
                                const std::string& valueSpecification,
                                const std::string& variableName,
                                msrOctaveEntryVariable&
                                                   octaveEntryKindVariable,
                                S_msrSemiTonesPitchAndOctave&
                                                   lilypondFixedOctaveEntryVariable);

  protected:

    // constructors/destructor
    // ------------------------------------------------------

                          lilypondFixedOctaveEntryAtom (
                                const std::string& longName,
                                const std::string& shortName,
                                const std::string& description,
                                const std::string& valueSpecification,
                                const std::string& variableName,
                                msrOctaveEntryVariable&
                                                   octaveEntryKindVariable,
                                S_msrSemiTonesPitchAndOctave&
                                                   lilypondFixedOctaveEntryVariable);

    virtual               ~lilypondFixedOctaveEntryAtom ();

  public:

    // set and get
    // ------------------------------------------------------

    void                  setFixedOctaveEntryVariableValue (
                                const S_msrSemiTonesPitchAndOctave& value)
                                  { fSemiTonesPitchAndOctaveVariable = value; }

    S_msrSemiTonesPitchAndOctave&
                          getFixedOctaveEntryVariableValue () const
                                  { return fSemiTonesPitchAndOctaveVariable; }

    // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    msrOctaveEntryVariable&
                          fOctaveEntryKindVariable;

    S_msrSemiTonesPitchAndOctave&
                          fSemiTonesPitchAndOctaveVariable;
};
```

This class in instanciated this way:

```
void lpsr2lilypondOahGroup::initializeNotesOptions ()
{
  // ... ... ...

  subGroup->
    appendAtomToSubGroup (
      lilypondFixedOctaveEntryAtom::create (
        "fixed", "",
R"(Use fixed octave entry reference PITCH_AND_OCTAVE in the generated LilyPond code.
PITCH_AND_OCTAVE is made of a diatonic pitch and
an optional sequence of commas or single quotes.
It should be placed between double quotes if it contains single quotes, such as:
  -fixed "c''")",
        "PITCH_AND_OCTAVE",
        "fFixedOctaveEntrySemiTonesPitchAndOctave",
        fOctaveEntryVariable,
        fFixedOctaveEntrySemiTonesPitchAndOctave));

  // ... ... ...
}
```

Applying such an atom with a value is done this way:

```
void lilypondFixedOctaveEntryAtom::applyAtomWithValue (
  const std::string& theString,
  std::ostream&      os)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gEarlyOptions.getTraceEarlyOptions ()) {
    std::stringstream ss;

    ss <<
      "==> oahAtom is of type 'lilypondFixedOctaveEntryAtom'";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // theString contains the fixed reference note

  // set octave entry kind
  fOctaveEntryKindVariable.setOctaveEntryKind (
    msrOctaveEntryKind::kOctaveEntryFixed);

  // set fixed octave reference value
  setFixedOctaveEntryVariableValue (
    msrSemiTonesPitchAndOctave::createFromString (
      K_MF_INPUT_LINE_UNKNOWN_,
      theString));

  fSelected = true;
}
```

17.4.1 The `oahAtomStoringAValue` class

Purely virtual class `oahAtomStoringAValue`, defined in `src/oah/oahBasicTypes.h/.cpp`, is the base class for them:

```
class EXP oahAtomStoringAValue : public oahValueFittedAtom
{
  // ... ... ...

  public:

    // print
    // ------------------------------------------------------

    virtual void          printAtomWithVariableEssentials (
                            std::ostream& os,
                            int fieldWidth) const;
    virtual void          printAtomWithVariableEssentials (
                            std::ostream& os,
                            int fieldWidth) const;

    void                  print (std::ostream& os) const override;
    void                  printFull (std::ostream& os) const override;

    void                  printHelp (std::ostream& os) const override;

    virtual void          displayAtomWithVariableOptionsValues (
                            std::ostream& os,
                              int       valueFieldWidth) const override;

  protected:

    // protected fields
    // ------------------------------------------------------

    std::string           fValueSpecification;

    std::string           fVariableName;
};
```

The field `oahAtomStoringAValue::fSelected` is necessary because some value types do not have an obvious 'neutral' element. This is the case for a note's octave, `oahLengthUnitKindAtom` and `oahColorRGBAtom`, for example. It is not used for data structures such as sets, lists and vector, since this is indicated by their size.

`fSelected` is set in `set*Variable ()` methods, as in method `oahIntegerAtom::setIntegerVariable ()` in `src/oah/oahAtomsCollection.cpp`:

```
void oahIntegerAtom::setIntegerVariable (int value)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gEarlyOptions.getTraceEarlyOptions ()) {
    std::stringstream ss;

    ss <<
      "Setting option '" <<
      fetchNames () <<
      "' integer variable to '" <<
      value <<
      "'";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fIntegerVariable = value;

  fSelected = true;
}
```

17.4.2 The `oahBooleanAtom` special case

Class `oahBooleanAtom` has its own `fSelected` field, because is it not derived from class `oahAtomStoringAValue`: there isn't any value to be supplied in the command line, since `fBooleanVariable` is `false` by default and becomes `true` when the options is selected:

```
//______________________________________________________________________________
class EXP oahBooleanAtom : public oahAtom
{
/*
  an atom controlling a Bool variable variableName,
  but expecting no value to be supplied:
  the variable is false initially,
  and is set to true by the mere occurrence of the atom
*/

  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    std::string           fVariableName;
    Bool&                 fBooleanVariable;
};
```

17.4.3 Checking whether an option has been selected

This in done by calls to the `getSelected ()` methods.

For example, method `lpsr2lilypondTranslator::generateLilypondVersion ()` in `src/formats/lpsr//lpsr2lilypondTranslator.cpp` contains:

```
void lpsr2lilypondTranslator::generateLilypondVersion ()
{
  // LilyPond version
  Bool
    lilypondVersionHasBeenSet =
      gGlobalLpsr2lilypondOahGroup->
        getLilypondVersionAtom ()->
          getSelected ();

  std::string
    lilypondVersion =
      lilypondVersionHasBeenSet
        ? gGlobalLpsr2lilypondOahGroup->
            getLilypondVersion ()
        : gGlobalLpsr2lilypondOahGroup->
           getLilypondVersionMinimumValue ();

  fLilypondCodeStream <<
    "\\version \"" <<
    lilypondVersion <<
    "\"" <<
    std::endl << std::endl;
}
```

The minimum LilyPond version number is `2.24.0`. Another can be selected with the `-lilypond-version`, `-lpv` option:

```
jacquesmenu@macmini > xml2ly -find lilypond-version
1 occurrence of string "lilypond-version" has been found:
   1:
    -lilypond-version, -lpv
    Set the Lilypond '\version' to STRING in the Lilypond code.
                    The default is '2.24.0'.
```

17.4.4 The oahAtomStoringAValue sub-classes

The classes derived from `oahAtomStoringAValue` are:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grh 'public oahAtomStoringAValue'
oah/oahHarmoniesExtraOah.h:class EXP extraDisplayAllHarmoniesStructuresAtom : public
    oahAtomStoringAValue
oah/oahHarmoniesExtraOah.h:class EXP extraDisplayAllHarmoniesContentsAtom : public
    oahAtomStoringAValue
oah/oahHarmoniesExtraOah.h:class EXP extraDisplayHarmonyDetailsAtom : public
    oahAtomStoringAValue
oah/oahHarmoniesExtraOah.h:class EXP extraDisplayHarmonyAnalysisAtom : public
    oahAtomStoringAValue
```

```
oah/oahAtomsCollection.h:class EXP oahIntegerAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahFloatAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahStringAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahRationalAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahNaturalNumbersSetElementAtom : public
    oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahColorRGBAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahIntSetElementAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahStringSetElementAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahStringToIntMapElementAtom : public
    oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahStringAndIntegerAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahStringAndTwoIntegersAtom : public
    oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahLengthUnitKindAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahLengthAtom : public oahAtomStoringAValue
oah/oahAtomsCollection.h:class EXP oahMidiTempoAtom : public oahAtomStoringAValue
formatsgeneration/brailleGeneration/brailleGenerationOah.h:class EXP brailleOutputKindAtom
     : public oahAtomStoringAValue
formatsgeneration/brailleGeneration/brailleGenerationOah.h:class EXP brailleUTFKindAtom :
    public oahAtomStoringAValue
formatsgeneration/brailleGeneration/brailleGenerationOah.h:class EXP
    brailleByteOrderingKindAtom : public oahAtomStoringAValue
formatsgeneration/msrGeneration/msrGenerationBasicTypes.h:class EXP
    msrGenerationAPIKindAtom : public oahAtomStoringAValue
formatsgeneration/multiGeneration/mfMultiGenerationOah.h:class EXP
    mfMultiGenerationOutputKindAtom : public oahAtomStoringAValue
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondScoreOutputKindAtom : public
    oahAtomStoringAValue
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondTransposePartNameAtom : public
    oahAtomStoringAValue
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondTransposePartIDAtom : public
    oahAtomStoringAValue
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondAccidentalStyleKindAtom : public
     oahAtomStoringAValue
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondChordsDisplayAtom : public
    oahAtomStoringAValue
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondLyricsNotesDurationsKindAtom :
    public oahAtomStoringAValue
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP
    lilypondDynamicsTextSpannersStyleKindAtom : public oahAtomStoringAValue
passes/lpsr2lilypond/lpsr2lilypondOah.h:class EXP lilypondBreakPageAfterMeasureNumberAtom
    : public oahAtomStoringAValue
passes/msr2lpsr/msr2lpsrOah.h:class EXP msr2lpsrScoreOutputKindAtom : public
    oahAtomStoringAValue
passes/msr2msr/msr2msrOah.h:class EXP msrIgnorePartAtom : public oahAtomStoringAValue
passes/msr2msr/msr2msrOah.h:class EXP msrKeepPartAtom : public oahAtomStoringAValue
passes/mxsr2msr/mxsr2msrOah.h:class EXP msrReplaceClefAtom : public oahAtomStoringAValue
formats/bsr/bsrOah.h:class EXP bsrFacSimileKindAtom : public oahAtomStoringAValue
formats/bsr/bsrOah.h:class EXP bsrTextsLanguageAtom : public oahAtomStoringAValue
formats/lpsr/lpsrOah.h:class EXP lpsrPitchesLanguageAtom : public oahAtomStoringAValue
formats/lpsr/lpsrOah.h:class EXP lpsrChordsLanguageAtom : public oahAtomStoringAValue
formats/lpsr/lpsrOah.h:class EXP lpsrTransposeAtom : public oahAtomStoringAValue
formats/msdl/msdlInputOah.h:class EXP msdlKeywordsLanguageAtom : public
    oahAtomStoringAValue
formats/msdl/msdlInputOah.h:class EXP msdlCommentsTypeAtom : public oahAtomStoringAValue
formats/msdl/msdlInputOah.h:class EXP msdlUserLanguageAtom : public oahAtomStoringAValue
formats/msdl/msdlInputOah.h:class EXP msdlPitchesLanguageAtom : public
    oahAtomStoringAValue
formats/msr/msrOah.h:class EXP msrPitchesLanguageAtom : public oahAtomStoringAValue
formats/msr/msrOah.h:class EXP msrRenamePartAtom : public oahAtomStoringAValue
```

17.5 Pure help atoms

Some options, such as -a, -about, only provide help to the user. Such pure help atoms can be with or without a value.

17.5.1 Pure help atoms without a value

The base oahPureHelpValueLessAtom class is defined in src/oah/oahBasicTypes.h/.cpp:

```
//______________________________________________________________________________
class EXP oahPureHelpValueLessAtom : public oahAtom
{
  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    std::string           fHelpValueLessAtomServiceName;
};
```

The actual pure help atoms without a value are:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grh 'public oahPureHelpValueLessAtom'
oah/oahAtomsCollection.h:class EXP oahOptionsUsageAtom : public oahPureHelpValueLessAtom
oah/oahAtomsCollection.h:class EXP oahHelpAtom : public oahPureHelpValueLessAtom
oah/oahAtomsCollection.h:class EXP oahHelpSummaryAtom : public oahPureHelpValueLessAtom
oah/oahAtomsCollection.h:class EXP oahAboutAtom : public oahPureHelpValueLessAtom
oah/oahAtomsCollection.h:class EXP oahVersionAtom : public oahPureHelpValueLessAtom
oah/oahAtomsCollection.h:class EXP oahLibraryVersionAtom : public oahPureHelpValueLessAtom
oah/oahAtomsCollection.h:class EXP oahHistoryAtom : public oahPureHelpValueLessAtom
oah/oahAtomsCollection.h:class EXP oahLibraryHistoryAtom : public oahPureHelpValueLessAtom
oah/oahAtomsCollection.h:class EXP oahContactAtom : public oahPureHelpValueLessAtom
```

17.5.2 Pure help value-fitted atoms

The base oahPureHelpValueFittedAtom class is defined in src/oah/oahBasicTypes.h/.cpp:

```
class EXP oahPureHelpValueFittedAtom : public oahValueFittedAtom
{
  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    std::string           fHelpValueFittedAtomServiceName; // JMI ???
};
```

The actual pure help value-fitted atoms are:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grh 'public oahPureHelpValueFittedAtom'
oah/oahAtomsCollection.h:class EXP oahQueryOptionNameAtom : public
    oahPureHelpValueFittedAtom
oah/oahAtomsCollection.h:class EXP oahFindStringAtom : public oahPureHelpValueFittedAtom
formats/msdl/msdlInputOah.h:class EXP oahDisplayMsdlKeywordsInLanguageAtom : public
    oahPureHelpValueFittedAtom
formats/msdl/msdlInputOah.h:class EXP oahDisplayMsdlTokensInLanguageAtom : public
    oahPureHelpValueFittedAtom
```

17.6 Options implicitly storing a value

There are options in multiGeneration to select the generated output :

```
jacquesmenu@macmini > msdlconverter --help-generate-output
--- Help for subgroup "Generated output" in group "Generated output group" ---
  Generated output group (-help-generate-output-group, -hgc-group):
  --------------------------
    Generated output (-help-generate-output, -hgo):
      -guido
            Generate Guido code as output.
      -lilypond
            Generate LilyPond code as output.
      -braille
            Generate braille code as output.
      -musicxml
            Generate MusicXML code as output.
```

No value is supplied in the command line or in a type `oahOptionsVector`, but a variable is used to store a value alright.

Purely virtual class `oahAtomImplicitlySettingAVariable` is the base class for this:

```
class EXP oahAtomImplicitlySettingAVariable : public oahAtom
/*
  a purely virtual common ancestor for all atom classes
  that store a value in a variable
  without taking it from argv or an oahOptionsVector
*/
{
  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    std::string           fVariableName;
};
```

This is used by class `mfMultiGenerationOutputKindAtom` defined in
`src/formatsgeneration/multiGeneration/multiGenerationOah.h/.cpp`:

```
class EXP mfMultiGenerationOutputKindAtom : public oahAtomStoringAValue
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    mfMultiGenerationOutputKind&
                          fMultiGenerationOutputKindVariable;
};
```

The value is stored in the variable in
constructor `mfMultiGenerationOutputKindAtom::mfMultiGenerationOutputKindAtom ()`:

```
mfMultiGenerationOutputKindAtom::mfMultiGenerationOutputKindAtom (
  const std::string&        longName,
  const std::string&        shortName,
  const std::string&        description,
  const std::string&        variableName,
  mfMultiGenerationOutputKind& mfMultiGenerationOutputKindVariable)
  : oahAtomImplicitlySettingAVariable (
      longName,
      shortName,
      description,
      variableName,
      oahElementValueKind::kElementValueLess),
    fMultiGenerationOutputKindVariable ( // this is where the value is supplied
      mfMultiGenerationOutputKindVariable)
{}
```

17.7 Options and help handling

- each option long name and non-empty short name must be unique in a given handler, to avoid ambiguities;
- an service `main ()` calls method `oahHandler::handleOptionsAndArgumentsFromArgcArgv ()`, in which:
- method `oahHandler::handleOptionNameCommon ()` handles the option names;
- `handleOptionValueOrArgument()`and the arguments to the service.
- contracted forms are expanded in method `oahHandler::handleOptionNameCommon ()` before the resulting, uncontracted options are handled;
- options handling works in two passes:
- the first one creates a list of class `oahElementUse` instances from `argc/argv` or an options and arguments;
- the second one traverses this list to apply the options that are used.
- the options are applied by virtual method `applyElement ()`, virtual method `applyAtomWithValue ()` and virtual method `applyAtomWithDefaultValue ()`;
- method `oahHandler::handleKnownArgvAtom ()` associatiates the value to the (preceding) field `oahHandler::f` if not null, or appends it to field `oahHandler::fHandlerArgumentsVector` to otherwise;
- `fPendingArgvValueFittedAtom` is used in `argv` contents handling to associate an option name with it value, which is the next element in `argv`.

17.8 Basic OAH types

They are defined in src/oah/oahBasicTypes.h/.cpp. The classes are:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/oah > grep class   oahBasicTypes.h
// PRE-declarations for class mutual dependencies
class   oahAtom;
class   oahSubGroup;
class   oahGroup;
class   oahHandler;
enum class oahOptionsDefaultValuesStyle {
enum class oahHandlerUsedThruKind {
class   oahPrefix;
class EXP oahPrefix : public smartable
  a common ancestor for all atom classes,
  this class   contains  only an uplink to the containing subgroup
class EXP oahAtom : public oahElement
```

```
/* this class is purely virtual
class EXP oahValueFittedAtom : public oahAtom
  a common ancestor for all atom classes
/* this class is purely virtual
  a common ancestor for all atom classes
class EXP oahAtomStoringAValue : public oahValueFittedAtom
/* this class is purely virtual
class EXP oahPureHelpValueLessAtom : public oahAtom
/* this class is purely virtual
class EXP oahPureHelpValueFittedAtom : public oahValueFittedAtom
/* this class is purely virtual
class EXP oahSubGroup : public oahElement
class EXP oahGroup : public oahElement
class EXP oahHandler : public smartable
 /* this class is purely virtual
enum class oahOptionalValuesStyleKind {
class EXP oahValueLessAtom : public oahAtom
/ * this class is purely virtual
```

17.9 Prefixes handling

17.10 `argc/argv` versus `oahOptionsVector`

Passing the options and arguments over to the library when using MusicFormats can be done in two ways:

- command line services get them from `argc/argv` as usual;
- application using the library through the API should place them in an `oahOptionsVector`, defined in `src/mflibrarymfMusicformatsErrors.h`:

```
typedef std::vector<std::pair<std::string, std::string> > oahOptionsVector;
```

Using an `oahOptionsVector` can be done for example:

- in Web sites;
- in the generators CLI services found in the `src/clisamples/` folder `src/clisamples/xml2Any.cpp`, `src/clisamples/libMultipleInitsTest.cpp`, `src/clisamples/Mikrokosmos3Wandering.cpp` and `src/clisamples/LilyPondIssue34.cpp`, as well as in `src/clisamples/msdl.cpp`, the MSDL converter command line interface.

In these services, an `oahOptionsVector` is instantiated ans populated from `argc/argv` with `convertArgcArgvToOptio`
defined in `src/oah/oahBasicTypes.h`:

```
EXP Bool convertArgcArgvToOptionsAndArguments (
  int                argc,
  char               *argv[],
  oahOptionsVector& theOptionsVector)
```

class `oahHandler` in `src/oah/oahBasicTypes.h/.cpp` contains:

```
    // options and arguments handling
    oahElementHelpOnlyKind
                            handleOptionsFromOptionsAndArguments (
                              std::string             serviceName,
                              const oahOptionsVector& theOptionsVector);

    oahElementHelpOnlyKind
                            handleOptionsAndArgumentsFromArgcArgv (
                              int    argc,
                              char* argv[]);

    virtual void            checkOptionsAndArgumentsConsistency ();

    virtual void            checkOptionsAndArguments () const = 0;
```

17.11 Applying options

Each oahElement, defined in src/oah/oahElements.h/.cpp, has an applyElement method:

```
    virtual void            applyValueLessAtom (std::ostream& os) = 0;
```

Atoms that can have an associated value are described in src/oah/oahBasicTypes.h/.cpp by class oahValueFittedA
which has methods applyAtomWithValue and applyAtomWithDefaultValue:

```
    virtual void            applyAtomWithValue (
                              const std::string& theString,
                              std::ostream& os) = 0;

    virtual void            applyAtomWithDefaultValue (std::ostream& os);
```

There are two methods for that:

```
    void                    applyValueLessAtom (std::ostream& os) override; %%%JMI
```

```
```

The last option is checked by method oahHandler::checkMissingPendingArgvValueFittedAtomValue () in src/oah/oahBasicTypes.cpp.

17.12 A OAH atoms collection

Frequent OAH atoms have been grouped in src/oah/oahAtomsCollection.h/.cpp. They are:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/oah > grep class   oahAtomsCollection.h
class EXP oahValueLessAtomAlias : public oahAtom
class EXP oahMacroAtom : public oahAtom
class EXP oahOptionsUsageAtom : public oahPureHelpValueLessAtom
class EXP oahHelpAtom : public oahPureHelpValueLessAtom
class EXP oahHelpSummaryAtom : public oahPureHelpValueLessAtom
class EXP oahAboutAtom : public oahPureHelpValueLessAtom
class EXP oahVersionAtom : public oahPureHelpValueLessAtom
class EXP oahContactAtom : public oahPureHelpValueLessAtom
class EXP oahBooleanAtom : public oahAtom
class EXP oahTwoBooleansAtom : public oahBooleanAtom
class EXP oahThreeBooleansAtom : public oahBooleanAtom
class EXP oahCombinedBooleansAtom : public oahAtom
class EXP oahCommonPrefixBooleansAtom : public oahAtom
class EXP oahIntegerAtom : public oahAtomStoringAValue
class EXP oahTwoIntegersAtom : public oahIntegerAtom
class EXP oahFloatAtom : public oahAtomStoringAValue
class EXP oahStringAtom : public oahAtomStoringAValue
class EXP oahFactorizedStringAtom : public oahAtom
class EXP oahDefaultedStringAtom : public oahStringAtom
class EXP oahRationalAtom : public oahAtomStoringAValue
class EXP oahNaturalNumbersSetElementAtom : public oahAtomStoringAValue
class EXP oahColorRGBAtom : public oahAtomStoringAValue
class EXP oahIntSetElementAtom : public oahAtomStoringAValue
class EXP oahStringSetElementAtom : public oahAtomStoringAValue
class EXP oahStringToIntMapElementAtom : public oahAtomStoringAValue
class EXP oahStringAndIntegerAtom : public oahAtomStoringAValue
class EXP oahStringAndTwoIntegersAtom : public oahAtomStoringAValue
class EXP oahLengthUnitKindAtom : public oahAtomStoringAValue
class EXP oahLengthAtom : public oahAtomStoringAValue
class EXP oahMidiTempoAtom : public oahAtomStoringAValue
class EXP oahOptionNameHelpAtom : public oahDefaultedStringAtom
class EXP oahQueryOptionNameAtom : public oahPureHelpValueFittedAtom
class EXP oahFindStringAtom : public oahPureHelpValueFittedAtom
```

See chapter 39 [The OAH atoms collection], page 283, for more details.

17.13 An option and help example

Option `-beam-all-grace-notes` controls whether beams should be added to grace notes. Here is how it is implemented and used.

First, we must determine to which internal representation or conversion pass it is applied to. In this case, that is the conversion pass of an MXSR to MSR. Thus we have in `src/passes/mxsr2msr/mxsr2msrOah.h`:

```
class EXP mxsr2msrOahGroup : public oahGroup

    Bool                      fBeamAllGraceNotes;

    Bool                      getBeamAllGraceNotes () const
                                  { return fBeamAllGraceNotes; }
```

In `src/passes/mxsr2msr/mxsr2msrOah.cpp`, the option is created this way:

```
void mxsr2msrOahGroup::initializeNotesOptions ()

  // beam all grace notes
  // --------------------------------------

  fBeamAllGraceNotes = false;

  S_oahBooleanAtom
    beamAllGraceNotesAtom =
      oahBooleanAtom::create (
        "beamagn", "beam-all-grace-notes",
R"(Add a beam to all grace notes)",
        "beamAllGraceNotes",
        fBeamAllGraceNotes);

  subGroup->
    appendAtomToSubGroup (
      beamAllGraceNotesAtom);
```

And that's it.

The option value is checked in `src/passes/mxsr2msr/mxsr2msrTranslator.cpp.h/.cpp`:

```
void mxsr2msrTranslator::visitStart ( S_grace& elt )

  // should all grace notes be beamed?
  if (gGlobalMxsr2msrOahGroup->getBeamAllGraceNotes ()) {
    fCurrentGraceIsBeamed = true;
  }

void mxsr2msrTranslator::handleStandaloneOrDoubleTremoloNoteOrGraceNoteOrRest (
  const S_msrNote& newNote)

      // create grace notes group
      fPendingGraceNotesGroup =
        msrGraceNotesGroup::create (
          inputLineNumber,
          msrGraceNotesGroupKind::kGraceNotesGroupBefore, // default value
          fCurrentGraceIsSlashed,
          fCurrentGraceIsBeamed,
          fCurrentMeasureNumber);
```

17.14 Options and help introspection

OAH represents options and the associated help in a tree of groups containing subgroups containing atoms. Searching it is easy, and there are options `-query` and `-find` for that.

Option '-query' provides informations about an option name:

```
jacquesmenu@macmini > xml2ly -query cpu
--- Help for atom "cpu" in subgroup "Informations"
    -cpu, -display-cpu-usage
          Write information about CPU usage to standard error.
```

Option `-find` searches the OAH tree ignoring letter cases:

```
jacquesmenu@macmini > xml2ly -find grace
5 occurrences of string "grace" have been found:
   1:
    -hgraces-group, -help-grace-notes-group

   2:
    -hgraces, -help-grace-notes

   3:
    -slashagn, -slash-all-grace-notes
    Add a slash to all grace notes
   4:
    -sluragn, -slur-all-grace-notes
    Add a slur to all grace notes
   5:
    -beamagn, -beam-all-grace-notes
    Add a beam to all grace notes
```

These two options are implementeds as `oahQueryOptionNameAtom` and `oahFindStringAtom` respectively in `src/oah/oahAtomsCollection.h/.cpp`.

17.15 Insider versus regular handlers

MusicFormats features two 'views' of the options and help available:

- the full view, named 'insider', contains the options grouped by format or pass;
- the default user view, named 'regular', contains options grouped by topic, such as tuplets or MIDI.

The 'regular' views have been introduced because there are many options and is was cumbersome to browse them without a user-oriented view by topics.

class `oahRegularHandler` relies on the corresponding insider handler:

```
  protected:

    // protected fields
    // ------------------------------------------------------

    S_oahHandler          fInsiderHandler;
```

A regular handler merely selects options from the `fInsiderHandler`, and presents them in groups and subgroups its own way. The group names are hidden to the user, and only the subgroups are seen in the help provided to the user.

For example, in `src/clisamples/xml2xml/xml2xmlRegularHandler.cpp`, there is:

```
void xml2xmlRegularHandler::createInformationsRegularGroup ()
{
  // group

  S_oahGroup
    group =
      oahGroup::create (
        "Informations group",
        "help-informations-group", "hinfos-group",
        "",
        oahElementVisibilityKind::kElementVisibilityWhole);
  appendGroupToRegulalHandler (group);

  // subgroup

  S_oahSubGroup
    subGroup =
      oahSubGroup::create (
        "Informations",
        "help-informations", "hinfos",
        "",
        oahElementVisibilityKind::kElementVisibilityWhole,
        group);
  group->
    appendSubGroupToGroup (subGroup);

  // atoms from the insider handler

  registerAtomInRegularSubgroup ("about", subGroup);
  registerAtomInRegularSubgroup ("version", subGroup);
  registerAtomInRegularSubgroup ("version-full", subGroup);
  registerAtomInRegularSubgroup ("history", subGroup);
  registerAtomInRegularSubgroup ("mf-version", subGroup);
  registerAtomInRegularSubgroup ("mf-history", subGroup);
  registerAtomInRegularSubgroup ("contact", subGroup);
}
```

An insider handler is always created, and a regular one relying on it is created if relevant. Here is how this is done this way, here in src/clisamples/msdl.cpp:

```
    // create an msdlConverter insider OAH handler
    // ------------------------------------------------------

    const S_msdlConverterInsiderHandler&
      insiderOahHandler =
        msdlConverterInsiderHandler::create (
          serviceName,
          serviceName + " insider OAH handler with argc/argv",
          multiGenerationOutputKind);

    // the OAH handler to be used, a regular handler is the default
    // ------------------------------------------------------

    if (insiderOption) {
      // use the insider msdlConverter OAH handler
      handler = insiderOahHandler;
    }
    else {
      // create a regular msdlConverter OAH handler
      handler =
        msdlConverterRegularHandler::create (
          serviceName,
          serviceName + " regular OAH handler with argc/argv",
          insiderOahHandler,
          multiGenerationOutputKind);
    }
```

17.16 Deciphering the options and arguments

17.16.1 Options and arguments multi-pass analysis

The options and arguments are first placed in a `mfOptionsAndArguments` instance:

- the command line services do this with interface function `convertArgcArgvToOptionsAndArguments ()` in their function, for example in src/clisamples/Mikrokosmos3Wandering.cpp:

```
int main (int argc, char*  argv[])
//------------------------------------------------------------------------
{
  // ... ... ...

  // create the global run data
  // ------------------------------------------------------

  gServiceRunData =
    mfServiceRunData::create (serviceName);

  // ... ... ...
}
```

- the API functions receive an `mfOptionsAndArguments` as an argument, here in src/converters/musicxml2musi

```
EXP mfMusicformatsErrorKind musicxmlfile2musicxml (
  const char*              fileName,
  mfOptionsAndArguments& handlerOptionsAndArguments,
  std::ostream&            out,
  std::ostream&            err)
{
  SXMLFile
```

```
    sxmlfile =
      createSXMLFileFromFile (
        fileName,
      mfPassIDKind::kMfPassID_1,
        "Create an MXSR from a MusicXML file");

  if (sxmlfile) {
    return
      xmlFile2musicxmlWithOptionsAndArguments (
        sxmlfile,
        handlerOptionsAndArguments,
        out,
        err);
  }

  return mfMusicformatsErrorKind::kMusicformatsErrorInvalidFile;
}
```

This is done using a two-pass scheme:

- first, a list of the options uses is built;
- then, the options and their arguments, if any, im this list are applied.

class `oahHandler` contains:

```
    // elements uses
    std::list<S_oahElementUse> fElementUsesList;

    // atoms waiting for a value
    S_oahValueFittedAtom
                          fPendingArgvValueFittedAtom;
    std::string           fNameUsedForPendingArgvValueFittedAtom;
```

17.16.2 Pure help runs

A pure help run is one in which MusicFormats in which help, without any other option. In such a case the run quit silently, otherwise it proceeds to performs its task. The type describing that is enumeration type `oahElementHelpOnlyKind`, defined in `src/oah/oahElements.h`:

```
enum class oahElementHelpOnlyKind {
  kElementHelpOnlyYes,
  kElementHelpOnlyNo
};
```

17.16.3 Applying options

The options are applied in `src/oah/oahBasicTypes.cpp` by method `oahHandler::applyOptionsFromElementUsesLi` defined in `src/oah/oahBasicTypes.h/.cpp`:

```
oahElementHelpOnlyKind oahHandler::applyOptionsFromElementUsesList ()
```

The heart of it is:

```
oahElementHelpOnlyKind oahHandler::applyOptionsFromElementUsesList ()
{
  // ... ... ...

        // the heart of it
        if (
          // group?
          S_oahGroup
            group =
              dynamic_cast<oahGroup*>(&(*elementUsed))
        ) {
          group->
            applyValueLessAtom (
              gOutput);
        }

        else if (
          // subgroup?
          S_oahSubGroup
            subGroup =
              dynamic_cast<oahSubGroup*>(&(*elementUsed))
        ) {
          subGroup->
            applyValueLessAtom (
              gOutput);
        }

        else {
          // this is an atom

          S_oahAtom
            atom =
              dynamic_cast<oahAtom*>(&(*elementUsed));

          oahElementValueKind
            atomValueKind =
              atom->
                getElementValueKind ();

          if (
            // value-fitted atom?
            S_oahValueFittedAtom
              valueFittedAtom =
                dynamic_cast<oahValueFittedAtom*>(&(*elementUsed))
          ) {
            switch (atomValueKind) {
              case oahElementValueKind::kElementValueLess:
                {
                  std::stringstream ss;

                  ss <<
                    "Atom with value " <<
                    valueFittedAtom->fetchNamesBetweenQuotes () <<
                    " has been registered as without value";

                  oahInternalError (ss.str ());
                }
                break;

%               case oahElementValueKind::kElementValueImplicit:
%                 valueFittedAtom->
%                   applyAtomWithDefaultValue (
%                     gOutput);
%                 break;

              case oahElementValueKind::kElementValueFitted:
                if (valueUsed.size ()) {
                  valueFittedAtom->
                    applyAtomWithValue (
                      valueUsed,
                      gOutput);
                }
                else {
                  std::stringstream ss;

                  ss <<
                    "Value-fitted atom " <<
                    valueFittedAtom->fetchNamesBetweenQuotes () <<
                    " needs a non-empty value";

                  oahInternalError (ss.str ());
                }
                break;

              case oahElementValueKind::kElementValueDefaulted:
                if (valueUsed.size ()) {
                  valueFittedAtom->
                    applyAtomWithValue (
                      valueUsed,
                      gOutput);
                }
                else {
                  valueFittedAtom->
                    applyAtomWithDefaultValue (
                      gOutput);
                }
                break;
            } // switch
          }

          else {
#ifdef MF_TRACE_IS_ENABLED
            if (gEarlyOptions.getTraceEarlyOptions ()) {
              gLog <<
                "**** Handling value-less atom:" <<
                std::endl;

              ++gIndenter;

              gLog <<
                atom <<
                std::endl;

              --gIndenter;
            }
#endif // MF_TRACE_IS_ENABLED

            atom->
              applyValueLessAtom (
                gOutput);
          }
        }

        // has a help-only been applied?
        switch (elementUsed->getElementHelpOnlyKind ()) {
          case oahElementHelpOnlyKind::kElementHelpOnlyYes:
            // a help option has been applied
            this->
              setOahHandlerFoundAHelpOption (
                elementUsed->
                  fetchNamesBetweenQuotes ());
            break;
          case oahElementHelpOnlyKind::kElementHelpOnlyNo:
            break;
        } // switch
      }

      else {
        std::stringstream ss;

        ss <<
          "Element from the from the atom uses list for \"" <<
          nameUsed <<
          "\" is null";

        oahInternalError (ss.str ());

  // ... ... ...
}
```

17.16.4 Early handling of some options

Debugging OAH needs the trace handling trace options option `-trace-oah, -toah` and option `-trace-oah-details,` `-toahd` to be activated first, even if they are not the first ones supplied.

The same holds to handle the option `-insider` option, since it involves using the insider handler and not a regular one.

Also, the `-quiet, -q` option should be handled early, to avoid errors in the options being reported.

Another case is the option `-trace-components, -tcomps` option. The versions data should exist when the option `-version, -v` is applied in method `oahVersionAtom::applyElement ()` in `src/oah/oahAtomsCollection.c` but building them should be able to produce a trace if this option is selected. This circularity circularity should thus be broken. Version history handling is presented in chapter **??** [musicformats components], page **??**.

The early options names are declared in `src/oah/oahEarlyOptions.h`:

```
// ... ... ...

// insider
EXP extern const std::string K_INSIDER_OPTION_LONG_NAME;
EXP extern const std::string K_INSIDER_OPTION_SHORT_NAME;

// ... ... ...

class EXP oahEarlyOptions
{
  // ... ... ...

  public:

    // set and get
    // ------------------------------------------------------

    void                  setEarlyInsiderOption ();
    Bool                  getEarlyInsiderOption () const
                              { return fEarlyInsiderOption; }

  // ... ... ...

 private:

    // fields
    // ------------------------------------------------------

    Bool                    fEarlyInsiderOption;

    // ... ... ...
}
```

Then, in src/oah/oahEarlyOptions.cpp, there is:

```
// ... ... ...

const std::string K_INSIDER_OPTION_LONG_NAME  = "insider";
const std::string K_INSIDER_OPTION_SHORT_NAME = "ins";

void oahEarlyOptions::setEarlyInsiderOption ()
{
  if (fTraceEarlyOptions) {
    std::stringstream ss;

    ss <<
      "Setting fEarlyInsiderOption" <<
      std::endl;
  }

  fEarlyInsiderOption = true;
}

// ... ... ...
```

Method `oahEarlyOptions::applyEarlyOptionIfRelevant ()` performs the analysis:

```
void oahEarlyOptions::applyEarlyOptionIfRelevant (
  const std::string& argumentWithoutDashToBeUsed,
  const std::string& optionValue)
{
  // this is OAH handling pass 1
  if (
    isEarlyOptionRecognized (
      argumentWithoutDashToBeUsed, K_INSIDER_OPTION_LONG_NAME)
      ||
    isEarlyOptionRecognized (
      argumentWithoutDashToBeUsed, K_INSIDER_OPTION_SHORT_NAME)
  ) {
    setEarlyInsiderOption ();
  }

  // ... ... ...
}
```

17.17 Implementing the -find option

Class `oahFindableElement` in src/oah/oahElements.h/.cpp is the base class of all those that can be found with this introspection option:

```
class EXP oahFindableElement : public smartable
{
  public:

    // creation from MusicXML
    // ------------------------------------------------------

/* this class is purely virtual
    static SMARTP<oahFindableElement> create ();
*/
```

```

  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    virtual Bool          findStringInFindableElement (
                            const std::string&          lowerCaseString,
                            std::list<S_oahFindStringMatch>& foundMatchesList,
                            std::ostream&                    os) const = 0;

  public:

    // print
    // ------------------------------------------------------

    virtual std::string   asString () const = 0;

    virtual void          print (std::ostream& os) const = 0;

    virtual const std::string   containingFindableElementAsString () const = 0;

  private:

    // private fields
    // ------------------------------------------------------
};
```

When matches are found, there are stored in a list of instances of `oahFindStringMatch`:

```
class   oahFindStringMatch : public smartable
{
  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    std::string           fFoundString;
    std::string           fContainingFindableElementInfo;
};
```

The `fContainingFindableElementInfo` describes the OAH element that the string was found in, either in its option name(s) or in its description.

For example, method method `oahHandler::findStringInFindableElement ()` in `src/oah/oahElement.cpp` creates an instance and appends it to the list :

```
Bool oahHandler::findStringInFindableElement (
  const std::string&     lowerCaseString,
  std::list<S_oahFindStringMatch>& foundMatchesList,
  std::ostream&                    os) const
{
  Bool result;

  // .. .. ..

  // does this handler's header match?
  Bool headerMatches =
    mfStringToLowerCase (fHandlerHeader).find (lowerCaseString) != std::string::npos;

  // does this handler's description match?
  Bool descriptionMatches =
    mfStringToLowerCase (fHandlerDescription).find (lowerCaseString) != std::string::npos;

  // does this handler's usage match?
  Bool usageMatches =
    mfStringToLowerCase (fHandlerUsage).find (lowerCaseString) != std::string::npos;

  if (headerMatches || descriptionMatches || usageMatches) {
    std::stringstream ss;

    ss <<
      fHandlerHeader <<
      ' ' <<
      fHandlerDescription <<
      ' ' <<
      fHandlerUsage;

    // append the match to foundStringsList
    foundMatchesList.push_back (
      oahFindStringMatch::create (
        ss.str (),
        containingFindableElementAsString ()));

    result = true;
  }

  // do this handler's prefixes match?
  if (fHandlerPrefixesMap.size ()) {
    ++gIndenter;

    for (
      std::map<std::string, S_oahPrefix>::const_iterator i =
        fHandlerPrefixesMap.begin ();
      i != fHandlerPrefixesMap.end ();
      ++i
    ) {
      S_oahPrefix
        prefix = (*i).second;

      // does the prefix match?
      prefix->
        findStringInFindableElement (
          lowerCaseString,
          foundMatchesList,
          os);
    } // for

    --gIndenter;
  }

  // do this handler's groups match?
  if (fHandlerGroupsList.size ()) {
    ++gIndenter;

    for (S_oahGroup group : fHandlerGroupsList) {
      group->
        findStringInGroup (
          lowerCaseString,
          foundMatchesList,
          os);
    } // for

    --gIndenter;
  }

  return result;
}
```

The same holds for method `oahPrefix::findStringInFindableElement ()` in `src/oah/oahElements.cpp`.

17.18 Checking options consistency

The options groups have a `checkGroupOptionsConsistency ()` to check that the use of the options if this group are consistent.

For example:

```
void mxsr2msrOahGroup::checkGroupOptionsConsistency ()
{
}
```

17.19 Adding new options

In order to make a new option available, one should:

- choose a short name and possibly a long name for the option;
- choose an atom class in `src/oah/oahAtomsCollection.h/.cpp` or write a new one if needed;
- decide in which subgroup and group the option should be made available in an insider OAH group, and create the latter two if needed;
- create a suitable atom and append it to the desired OAH subgroup;
- check the use of the atom wherever needed in the code base;
- add the new atom's long name to the corresponding regular OAH group;
- and last but not least… test the result.

  This should be done first with the `-insider, -ins` option, and then without it to ensure that the regular OAH handler knows the new option too.

17.19.1 'Representations' vs. 'passes' options

When adding a new option, it is sometimes not clear whether to assign it to a format or to the passes that create or browse it.

For example, the trace of `<backup/>` and `<forward/>` is used by both `mxsr2msrOah` and `msr2mxsrOah`. The corresponding options are thus placed in `src/formats/mxsr/mxsrOah.h/.cpp`:

17.19.2 Using an existing OAH atom class

When option `-reverse-names-display-order, -rndo` was added to OAH by this author:

- class `oahBooleanAtom` was ready to be used;
- it was decided to place it in the global variable `gOahOahGroup` OAH group, in its `Options help` sub group;
- class `oahOahGroup` in src/oah/oahOah.h/.cpp got a new `fReverseNamesDisplayOrder` field:

```
class EXP oahOahGroup : public oahGroup
{
    void                      setReverseNamesDisplayOrder ()
                                  { fReverseNamesDisplayOrder = true; }
    Bool                      getReverseNamesDisplayOrder () const
                                  { return fReverseNamesDisplayOrder; }

    // ... ... ...

    Bool                      fReverseNamesDisplayOrder;

    // ... ... ...
};
```

- method `oahOahGroup::initializeOahBasicHelpOptions ()` was augmented with:

```
void oahOahGroup::initializeOahBasicHelpOptions (
  std::string serviceName)
{
  // ... ... ...

  // reverse names display order

  fReverseNamesDisplayOrder = false;

  subGroup->
    appendAtomToSubGroup (
      oahBooleanAtom::create (
        "rndo", "reverse-names-display-order",
R"(Write the short names before the long ones.)",
        "reverseNamesDisplayOrder",
        fReverseNamesDisplayOrder));

  // ... ... ...
}
```

- method `oahOahGroup::printOahOahValues ()` was augmented with:

```
void oahOahGroup::printOahOahValues (int valueFieldWidth)
{
  gLog <<
    "The basic options are:" <<
    std::endl;

  // ... ... ...

  // options and help display
  // --------------------------------------

  gLog << std::left <<
    std::setw (valueFieldWidth) << "Options trace and display:" <<
    std::endl;

  ++gIndenter;

  gLog << std::left <<
    std::setw (valueFieldWidth) << "fReverseNamesDisplayOrder" << ": " <<
    fReverseNamesDisplayOrder <<
    std::endl <<

  // ... ... ...
```

- then tests of the use of option `-reverse-names-display-order, -rndo` were added in `src/oah/oahElements.c` such as in method `oahElement::fetchNames ()`:

```
std::string oahElement::fetchNames () const
{
  std::stringstream ss;

  if (
    fShortName.size ()
        &&
    fLongName.size ()
  ) {
    if (gOahOahGroup->getReverseNamesDisplayOrder ()) {
      ss <<
        '-' << fShortName <<
        ", " <<
        '-' << fLongName;
    }
    else {
      ss <<
        '-' << fLongName <<
        ", " <<
        '-' << fShortName;
    }
  }

  else {
    if (fShortName.size ()) {
      ss <<
      '-' << fShortName;
    }
    if (fLongName.size ()) {
      ss <<
        '-' << fLongName;
    }
  }

  return ss.str ();
}
```

- and finally, all `*RegularHandler::createOahRegularGroup ()` methods were augmented with:

```
 void msdl2brailleRegularHandler::createOahRegularGroup ()
{
  // ... ... ...

  registerAtomInRegularSubgroup ("reverse-names-display-order", subGroup);

  // ... ... ...
}
```

17.19.3 Creating a new OAH atom class without a value

When class `oahHistoryAtom` was added to OAH, the first thing has been to add a `printHistory ()` in class `mfcMultiComponent` in `src/mfutilities/mfcBasicTypes.h`:

```
class   mfcMultiComponent : public smartable
{
  // ... ... ...

  public:

    // print
    // ------------------------------------------------------

    void                      print (std::ostream& os) const;
    void                      printHistory (std::ostream& os) const;

  // ... ... ...
};
```

Then the next thing has been to clone class `oahVersionAtom` in `src/oah/oahAtomCollection.h/.cpp`, renaming `printVersion ()` to `printHistory ()`:

```
//______________________________________________________________________________
class EXP oahHistoryAtom : public oahPureHelpValueLessAtom
{
  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    void                      applyValueLessAtom (std::ostream& os) override;

  public:

    // visitors
    // ------------------------------------------------------

    void                      acceptIn  (basevisitor* v) override;
    void                      acceptOut (basevisitor* v) override;

    void                      browseData (basevisitor* v) override;

    // print
    // ------------------------------------------------------

    void                      print (std::ostream& os) const override;

    void                      printHistory (std::ostream& os) const;
};
```

Then in method `oahHistoryAtom::printHistory ()`, the call to `printVersion ()` has been replaced by a call to `printHistory ()`:

```
void oahHistoryAtom::applyValueLessAtom (std::ostream& os)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gEarlyOptions.getTraceEarlyOptions ()) {
    std::stringstream ss;

    ss <<
      "==> option '" << fetchNames () << "' is a oahHistoryAtom";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  int saveIndent = gIndenter.getIndent ();

  gIndenter.resetToZero ();

  printHistory (os);

  gIndenter.setIndent (saveIndent);

  fSelected = true;
}
```

method :: () has the be adapted as:

```
void oahHistoryAtom::printHistory (std::ostream& os) const
{
  // get the handler history
  S_mfcMultiComponent
    handlerMultiComponent =
      fetchAtomUpLinkToHandler ()->
        getHandlerMultiComponent ();

#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  mfAssert (
    __FILE__, __LINE__,
    handlerMultiComponent != nullptr,
    "handlerMultiComponent is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  handlerMultiComponent->
    printHistory (os);
}
```

Then a new option has been added in method `oahOahGroup::initializeOahBasicHelpOptions ()`, in `src/oah/oahOah.cpp`:

```
  // history

  subGroup->
    appendAtomToSubGroup (
      oahHistoryAtom::create (
        "hist", "history",
        regex_replace (
R"(Display EXECUTABLE_NAME's history.)",
          std::regex ("EXECUTABLE_NAME"),
          serviceName),
        serviceName));
```

And the new option long name `version` has been added to all regular OAH handlers that already contained the option `-version, -v`, such as in method `xml2lyRegularHandler::createInformationsRegularGroup ()`, alongside the existing option `-version, -v`:

```
  registerAtomInRegularSubgroup ("version", subGroup);
  registerAtomInRegularSubgroup ("version-full", subGroup);
  registerAtomInRegularSubgroup ("history", subGroup);
  registerAtomInRegularSubgroup ("mf-version", subGroup);
  registerAtomInRegularSubgroup ("mf-history", subGroup);
```

17.19.4 Creating a new OAH value-fitted atom class

Let's look at how class `oahLengthAtom` is implemented.

class `msrLength` is defined in `src/formats/msr/msrBasicTypes.h/.cpp`:

```
class EXP msrLength : public smartable
{
  // ... ... ...

    // public services
    // ------------------------------------------------------

    Bool                      operator== (const msrLength& other) const
                                  {
                                    // JMI convert to same length unit kind before comparing
    BLARK
                                    return
                                      fLengthUnitKind == other.fLengthUnitKind
                                       &&
                                      fLengthValue == other.fLengthValue;
                                  }

    Bool                      operator!= (const msrLength& other) const
                                  { return ! ((*this) == other); }

    void                      convertToLengthUnit (
                                msrLengthUnitKind lengthUnitKind);

  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    msrLengthUnitKind     fLengthUnitKind;
    float                 fLengthValue;
};
```

Enumeration type is defined in `src/formats/msr/msrBasicTypes.h` as:

```
enum class msrLengthUnitKind {
  kUnitInch, kUnitCentimeter, kUnitMillimeter
};
```

Here is the declaration of class `oahLengthAtom` in `src/oah/oahAtomsCollection.h`:

```
class EXP oahLengthAtom : public oahAtomStoringAValue
{
/*
  an atom controlling a length variable
*/

  // ... ... ...

  public:

    // set and get
    // ------------------------------------------------------

    void                      setLengthVariable (
                                msrLength value)
                                  {
                                    fLengthVariable = value;
                                  }

    msrLength             getLengthVariable () const
                              { return fLengthVariable; }

  public:

    // public services
    // ------------------------------------------------------

    void                  applyAtomWithValue (
                            const std::string& theString,
                            std::ostream&      os) override;

  // ... ... ...

  public:

    // print
    // ------------------------------------------------------

  // ... ... ...

    void                  displayAtomWithVariableOptionsValues (
                            std::ostream& os,
                            int           valueFieldWidth) const override;

  private:

    // private fields
    // ------------------------------------------------------

    msrLength&            fLengthVariable;
};
```

Method **oahLengthAtom::applyAtomWithValue ()** in `src/oah/oahAtomsCollection.cpp` deciphers the string supplied by the user and stores it the class **msrLength** variable:

```
void oahLengthAtom::applyAtomWithValue (
  const std::string& theString,
  std::ostream&      os)
{
  // ... ... ...

  std::regex e (regularExpression);
  std::smatch sm;

  regex_match (theString, sm, e);

  unsigned int smSize = sm.size ();

  // ... ... ...

  if (smSize == 4) {
    // leave the low level details to the STL...
    float floatValue;
    {
      std::stringstream ss;
      // concatenate the integer and decimal parts
      ss << sm [ 1 ] << sm [ 2 ];
      ss >> floatValue;
    }

    std::string lengthUnitName = sm [ 3 ];

    // is lengthUnitName known in the length unit names map?
    std::map<std::string, msrLengthUnitKind>::const_iterator
```

```
      it =
        gGlobalMsrLengthUnitKindsMap.find (
          lengthUnitName);

    if (it == gGlobalMsrLengthUnitKindsMap.end ()) {
      // no, length unit name is unknown in the map

      std::stringstream ss;

      ss <<
        "length unit name \"" << lengthUnitName <<
        "\" is unknown" <<
        std::endl <<
        "The " <<
        gGlobalMsrLengthUnitKindsMap.size () <<
        " known length unit names are:" <<
        std::endl;

      ++gIndenter;

      ss <<
        availableMsrLengthUnitKinds (K_MF_NAMES_LIST_MAX_LENGTH);

      --gIndenter;

      oahError (ss.str ());
    }

    setLengthVariable (
      msrLength (
        (*it).second,
        floatValue));
  }

  else {
    std::stringstream ss;

    ss <<
      "length value \"" << theString <<
      "\" for option '" << fetchNames () <<
      "' is ill-formed";

    oahError (ss.str ());
  }

  fSelected = true;
}
```

Method `oahLengthAtom::displayAtomWithVariableOptionsValues ()` is in charge of displaying the length value when option `-display-options-values`dov is selected:

```
void oahLengthAtom::displayAtomWithVariableOptionsValues (
  std::ostream& os,
  int       valueFieldWidth) const
{
  os << std::left <<
    std::setw (valueFieldWidth) <<
    fVariableName <<
    " : " <<
    fLengthVariable.asString ();
  switch (fEarlyOptionKind) {
    case oahEarlyOptionKind::kEarlyOptionNo:
      break;
    case oahEarlyOptionKind::kEarlyOptionYes:
      os <<
        ", early";
      break;
  } // switch
  if (fSelected) {
    os << ", selected";
  }
  os << std::endl;
}
```

Then an option to set the LilyPond paper height can be added to the relevant OAH options group in method `lpsrOahGroup::initializeLpsrPaperOptions ()` in `src/formats/lpsr//lpsrOah.cpp` by:

```
  // paper height

  fPaperHeight.setLengthUnitKind (msrLengthUnitKind::kUnitMillimeter);
  fPaperHeight.setLengthValue (297);

  fPaperHeightAtom =
    oahLengthAtom::create (
      "paper-height", "",
R"(Set the LilyPond 'paper-height' paper variable to HEIGHT in the LilyPond code.
HEIGHT should be a positive floating point or integer number,
immediately followed by a unit name, i.e. 'in', 'mm' or 'cm'.
By default, LilyPond uses 297 mm (A4 format).)",
      "HEIGHT",
      "paperHeight",
      fPaperHeight);

  subGroup->
    appendAtomToSubGroup (
      fPaperHeightAtom);
```

17.20 Extra options

The description of music scores in MusicFormats is quite rich, and it was easy (and temptating…) to offer options such as:

```
jacquesmenu@macmini > xml2ly -query show-harmony-analysis
--- Help for atom "display-harmony-analysis" in subgroup "Harmony analysis" of group "
    Extra" ---
    -sca, -show-harmony-analysis HARMONY_SPEC
          Write an analysis of the harmony for the given diatonic (semitones) pitch
          in the current language and the given harmony to standard output.
          HARMONY_SPEC can be:
          'ROOT_DIATONIC_PITCH HARMONY_NAME INVERSION'
          or
          "ROOT_DIATONIC_PITCH = HARMONY_NAME INVERSION"
          Using double quotes allows for shell variables substitutions, as in:
          HARMONY="maj7"
          INVERSION=2
          xml2ly -show-harmony-analysis "bes ${HARMONY} ${INVERSION}"
```

This is done in `src/oah/harmoniesExtraOah.h/.cpp`. It suffices to call function `createGlobalHarmoniesExtraOahG`

```
  // create the extra OAH group
  appendGroupToHandler (
    createGlobalHarmoniesExtraOahGroup ());
```

Macro `MF_HARMONIES_EXTRA_IS_ENABLED` is defined or not in `src/oah/mfEnableHarmoniesExtraSetting.h`:

```
// comment the following definition if no extra options are wanted

#ifndef MF_HARMONIES_EXTRA_IS_ENABLED
  #define MF_HARMONIES_EXTRA_IS_ENABLED
```

17.21 man pages generation

MusicFormats can create man pages for its command line services by browing their OAH hierarchy. This has not been finalized yet.

17.22 Specific global OAH groups

Some informations need to be available globally in the MusicFormats library, such a the conversion date and command line. They are groupeds in src/oah/generalOah.h/.cpp:

```
class EXP generalOahGroup : public oahGroup
{
  // ... ... ...

  private:

    // translation date
    // --------------------------------------

    std::string           fTranslationDateFull;
    std::string           fTranslationDateYYYYMMDD;

    // warning and error handling
    // --------------------------------------

    Bool                  fQuiet;

    Bool                  fDontShowErrors;
    Bool                  fDontQuitOnErrors;

    Bool                  fDisplaySourceCodePositions;

    // CPU usage
    // --------------------------------------

    Bool                  fDisplayCPUusage;
};
```

There are also harmonies-specific options grouped in src/oah/harmoniesExtraOah.h/.cpp. They are available as icing on the cakeicing on the cake independently of any conversion activity:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/oah > grep class   oahHarmoniesExtraOah.h
class EXP extraDisplayAllHarmoniesStructuresAtom : public oahAtomStoringAValue
class EXP extraDisplayAllHarmoniesContentsAtom : public oahAtomStoringAValue
class EXP extraDisplayHarmonyDetailsAtom : public oahAtomStoringAValue
class EXP extraDisplayHarmonyAnalysisAtom : public oahAtomStoringAValue
class EXP harmoniesExtraOahGroup : public oahGroup
```

```
class EXP harmoniesExtraOahGroup : public oahGroup
{
  // ... ... ...

  public:

    // fields
    // --------------------------------------------------------

    std::string           fHarmoniesRootAsString;
};
```

17.23 Visiting OAH groups

As an internal representation, OAH can be browsed with the two-phase visitors. This is useful:

- to produce man pages automatically from the options available;
- to create the code that proposes the options to the user in a Web site, also automatically.

Part VII

Representations

Chapter 18

Representations general principles

18.1 Trees vs graphs, denormalization

In databases, denormalization means that some data is present in several places. This is usually done for speed, at the cost of making updates more complex, since no such place should be ignored in an update.

A music score can be represented as a tree of elements, but performing conversions of such representations needs shortcuts to be more practical. MusicFormats used the term link for that.

18.2 Denormalization

18.2.1 Descriptions sharing

MSR uses denormalization explicitly, with smart pointers to class instances being stored in other instances.

In particular, class `msrChord` contains elements that are actually detained by the notes it is composed of:

```
    // articulations
    std::list<S_msrArticulation>
                          fChordArticulationsList;

    // spanners
    std::list<S_msrSpanner>
                          fChordSpannersList;

    // single tremolo
    S_msrSingleTremolo    fChordSingleTremolo;
```

This is to avoid having to browse the chord's components to obtain the corresponding information each time it is needed.

All such denormalization is done in MSR internally: the code using MSR does not have to denormalize itself. It can use whichever occurrence of any given denormalized data safely, though.

18.2.2 Multiple voices

Another aspect to account for is that of dynamics, lyrics, harmonies and figured bass.

LilyPond supplies specific support to have them outside of notes, chords and other sound-producing score elements. This provides flexibility when combining a score's staves and voices in various ways depending on the needs .

MusicFormats has explicit voice kinds for this, declared in src/formats/msr/msrVoices.h:

```
enum class msrVoiceKind {
  kVoiceKindRegular,
  kVoiceKindDynamics,
  kVoiceKindHarmonies,  // for MusicXML <harmony/>, LilyPond ChordNames
  kVoiceKindFiguredBass // for MusicXML <figured-bass/>, LilyPond FiguredBass
};
```

In MSR, for example, a harmony is both attached to a note in a regular voice and an element of a harmony voice:

```
    std::list<S_msrHarmony>
                          fNoteHarmoniesList;
```

An `msrNote` instance will thus be browsed twice, when those two voices are. The ones attached to a note are browsed this way:

```
void msrNote::browseData (basevisitor* v)
{
  // ... ... ...

  // browse the harmonies if any
  if (fNoteHarmoniesList.size ()) {
    ++gIndenter;
    for (S_msrHarmony harmony : fNoteHarmoniesList) {
      // browse the harmony
      msrBrowser<msrHarmony> browser (v);
      browser.browse (*harmony);
    } // for
    --gIndenter;
  }

  // ... ... ...
};
```

18.3 Newborn clones

The multi-pass structure of the converters build with `musicformat` leads to a question: should an existing description, such as that of a barLine or a note, be used as is, or should it be built again?

Depending of the kind of description, both possibilities are used:

- the description is used as is if it is *shallow*, i.e. it doesn't contain smart-pointers to data – it is *self-contained*;
- otherwise, a new description is built, sharing some smart-pointers fieds with the existing one if needed. This newborn clone is then populated when it is is inserted in the representation being built.

For example, in src/passes/msr2lpsr/, the `S_msrBarLine` values found in the MSR data are used also in the LPSR data:

```
void msr2lpsrTranslator::visitStart (S_msrBarLine& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  int inputLineNumber =
    elt->getInputStartLineNumber ();
#endif // MF_TRACE_IS_ENABLED

#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrBarLine" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // ... ... ...

  // append the barLine to the current voice clone
  fCurrentVoiceClone->
    appendBarLineToVoice (elt);
}
```

On the opposite, a new `S_msrVoice` description is built for use by LPSR: this is how the LilyPond #34 issue is circumvented, adding skip notes where needed in the voices that don't have grace notes at their beginning.

Such new descriptions are created by `*NewbornClone ()` methods, such as:

```
S_msrTuplet msrTuplet::createTupletNewbornClone ()
{
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceTuplets ()) {
    std::stringstream ss;

    ss <<
      "Creating a newborn clone of tuplet " <<
      asString ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  S_msrTuplet
    newbornClone =
      msrTuplet::create (
        fInputStartLineNumber,
        fBarLineUpLinkToMeasure->getMeasureNumber (),
        fTupletNumber,
        fTupletBracketKind,
        fTupletLineShapeKind,
        fTupletShowNumberKind,
        fTupletShowTypeKind,
        fTupletFactor,
        fMemberNotesSoundingWholeNotes,
        fMemberNotesDisplayWholeNotes);

  return newbornClone;
}
```

Such a newborn clone is created and used this way in method :: ():

```
void msr2lpsrTranslator::visitStart (S_msrTuplet& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrTuplet" <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create the tuplet clone
  S_msrTuplet
    tupletClone =
      elt->createTupletNewbornClone ();

  // register it in this visitor
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceTuplets ()) {
    std::stringstream ss;

    ss <<
      "++> pushing tuplet '" <<
      tupletClone->asString () <<
      "' to tuplets stack";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fTupletClonesStack.push_front (tupletClone);

  // is Scheme support needed?
  switch (elt->getTupletLineShapeKind ()) {
    case msrTupletLineShapeKind::kTupletLineShapeStraight:
      break;
    case msrTupletLineShapeKind::kTupletLineShapeCurved:
      fResultingLpsr->
        // this score needs the 'tuplets curved brackets' Scheme function
        setTupletsCurvedBracketsSchemeFunctionIsNeeded ();
      break;
  } // switch
}
```

18.4 Deep clones

Some classes in MusicFormats, such as class `msrVoice` in src/formats/msr/msrVoices.h/.cpp, have a `*DeepClone ()` method:

```
    SMARTP<msrVoice> createVoiceDeepClone (
                          int          inputLineNumber,
                          msrVoiceKind voiceKind,
                          int          voiceNumber,
                          const S_msrStaff&   containingStaff);
```

Deep copies of the MSR data is not used currently. This can be changed should the need arise in the future.

18.5 Inheritance

18.5.1 Single inheritance

Many classes in MusicFormats use single inheritance. For example, in src/formats/msr/msrTimeSignature.h:

```
class EXP msrTimeSignature : public msrMeasureElement
{
  public:

    // creation from MusicXML
    // ------------------------------------------------------

    static SMARTP<msrTimeSignature> create (
                            int              inputLineNumber,
                            const S_msrMeasure&  upLinkToMeasure,
                            msrTimeSignatureSymbolKind
                                             timeSignatureSymbolKind);

    // creation from the applications
    // ------------------------------------------------------

    static SMARTP<msrTimeSignature> createTwoEightsTime (
                            int inputLineNumber);

    // ... ... ...

    // creation from the applications
    // ------------------------------------------------------

    static SMARTP<msrTimeSignature> createTimeFromString (
                            int    inputLineNumber,
                            std::string timeString);

    // ... ... ...
```

The definitions in in src/formats/msr/msrTimeSignature.cpp are:

```
S_msrTimeSignature msrTimeSignature::create (
  int            inputLineNumber,
  S_msrMeasure   upLinkToMeasure,
  msrTimeSignatureSymbolKind
                 timeSignatureSymbolKind)
{
  msrTimeSignature* obj =
    new msrTimeSignature (
      inputLineNumber,
      upLinkToMeasure,
      timeSignatureSymbolKind);
  assert (obj != nullptr);
  return obj;
}

msrTimeSignature::msrTimeSignature (
  int            inputLineNumber,
  S_msrMeasure   upLinkToMeasure,
  msrTimeSignatureSymbolKind
                 timeSignatureSymbolKind)
    : msrMeasureElement (
        inputLineNumber)
{
  fTimeSignatureSymbolKind = timeSignatureSymbolKind;

  fTimeIsCompound = false;
}
```

18.5.2 Single inheritance for smart pointers

All classes for which smart pointers are needed should inherit from class `smartable`, such as in `src/formats/msdl/msd`

```
class   msdlScanner : public smartable
{
  public:

    // creation
    // ------------------------------------------------------

    static SMARTP<msdlScanner> create (std::istream& inputStream);

  public:

    // constructors/destructor
    // ------------------------------------------------------

                              msdlScanner (std::istream& inputStream);

  // ... ... ...
};
```

This leads to the following in in `src/formats/msdl/msdlScanner.cpp`:

```
S_msdlScanner msdlScanner::create (std::istream& inputStream)
{
  msdlScanner* obj =
    new msdlScanner (inputStream);
  assert (obj != nullptr);
  return obj;
}

msdlScanner::msdlScanner (std::istream& inputStream)
    : fInputStream (
        inputStream),
      fCurrentToken (
        ),
      fCurrentTokenKind (
        fCurrentToken.getTokenKindNonConst ()),
      fCurrentTokenDescription (
        fCurrentToken.getTokenDescriptionNonConst ())
{
  // trace
#ifdef MF_TRACE_IS_ENABLED
  fTraceTokens        = gGlobalMsdl2msrOahGroup->getTraceTokens ();
  fTraceTokensDetails = gGlobalMsdl2msrOahGroup->getTraceTokensDetails ();
#endif // MF_TRACE_IS_ENABLED

  // ... ... ...
}
```

18.5.3 Multiple inheritance for visitors

Multiple inheritance is used extensively in visitors, which is the way to specify what elements are it seen by the visitor. For example, in src/formats/msr/msr2msrTranslator.h, there is:

```
class EXP msr2msrTranslator :

  public visitor<S_msrScore>,

  // rights

  public visitor<S_msrIdentification>,

  public visitor<S_msrCredit>,
  public visitor<S_msrCreditWords>,

    // ... ... ...
};
```

Then there are `visitStart ()` and/or `visitEnd ()` methods to handle the corresponding elements:

```
void msr2msrTranslator::visitStart (S_msrIdentification& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrIdentification" <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  ++gIndenter;

  // set the current identification
  fCurrentIdentification = elt;

  // store it in the resulting MSR score
  fResultingNewMsrScore->
    setIdentification (
      fCurrentIdentification);

  fOnGoingIdentification = true;
}
```

```
void msr2msrTranslator::visitEnd (S_msrIdentification& elt)
{
  fOnGoingIdentification = false;

  --gIndenter;

#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> End visiting msrIdentification" <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__,  __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED
}
```

Forgetting to define those `visit* ()` methods causes *no error message whatsoever*: the corresponding elements are just not handled by the visitor.
The visitors trace options are useful to detect such cases:

```
jacquesmenu@macmini: ~/musicformats-git-dev/musicxml > xml2ly -find visitors
3 occurrences of string "visitors" have been found:
   1:
    -tmxmltvis, -trace-mxsr-visitors
    Write a trace of the MusicXML tree visiting activity to standard error.
   2:
    -tmsrvis, -trace-msr-visitors
    Write a trace of the MSR graphs visiting activity to standard error.
   3:
    -tlpsrvis, -trace-lpsr-visitors
    Write a trace of the LPSR graphs visiting activity to standard error.
```

18.5.4 Multiple inheritance in other classes

The only such case is class `mfIndentedOstream` in src/utilities/mfIndentedTextOutput.cpp.h/.cpp:

```
class EXP mfIndentedOstream: public std::ostream, public smartable
{
/*
Reference for this class:
  https://stackoverflow.com/questions/2212776/overload-handling-of-stdendl

Usage:
  mfIndentedOstream myStream (std::cout);

  myStream <<
    1 << 2 << 3 << std::endl <<
    5 << 6 << std::endl <<
    7 << 8 << std::endl;
*/

  public:

    // creation
    // ------------------------------------------------------

    static SMARTP<mfIndentedOstream> create (
      std::ostream&       theOStream,
      mfOutputIndenter& theIndenter)
    {
      mfIndentedOstream* obj = new mfIndentedOstream (
        theOStream,
        theIndenter);
      assert (obj != nullptr);

      return obj;
    }

    // constructors/destructor
    // ------------------------------------------------------

                          mfIndentedOstream (
                            std::ostream&       theOStream,
                              mfOutputIndenter& theIndenter)
                              : std::ostream (
                                  & fIndentedStreamBuf),
                                fIndentedStreamBuf (
                                  theOStream,
                                  theIndenter)
                                {}

    virtual                 ~mfIndentedOstream () {};

  public:

    // public services
    // ------------------------------------------------------

    // flush
    void                    flush ()
                                { fIndentedStreamBuf.flush (); }

    // indentation
    mfOutputIndenter&         getIndenter () const
                                { return fIndentedStreamBuf.getOutputIndenter (); }

    void                    incrIndentation ()
                                { ++ (fIndentedStreamBuf.getOutputIndenter ()); }

    void                    decrIndentation ()
                                { -- (fIndentedStreamBuf.getOutputIndenter ()); }

  private:

    // private fields
    // ------------------------------------------------------

    // mfIndentedOstream just uses an mfIndentedOStreamBuf
    mfIndentedOStreamBuf      fIndentedStreamBuf;

};
typedef SMARTP<mfIndentedOstream> S_mfIndentedOstream;
```

18.5.5 Reversibility

All formats in MusicFormats that can be obtained by a conversion from another one should be convertible back in the latter, without information loss.
Thus:

- MXSR contains nearly everything that can be described in MusicXML data. The main std::exception at the time of this writing is the MIDI information, see subsection 26.1.1 [MusicXML coverage], page 237;
- MSR contains MusicXML-related informations, so as to convert it back to MXSR;
- LSPR and BSR contain an MSR component. This is why converting those formats back to MSR is merely getting the corresponding field.

Chapter 19

Displaying formats

MusicFormats is equipped with option `-display*` options as a help to the maintainer.

19.1 Display categories

19.2 Displaying in practise

```
%void lpsr2lilypondTranslator::generateCodeForNoteInMeasure (
%  const S_msrNote& note)
%{
%  int inputLineNumber =
%    note->getInputStartLineNumber ();
%
%#ifdef MF_TRACE_IS_ENABLED
%  if (gTraceOahGroup->getTraceNotes ()) {
%    std::stringstream ss;
%
%    ss <<
%      std::endl <<
%      "% --> generating code for noteRegularInMeasure " <<
%      note->asString () <<
%      ", line " << inputLineNumber <<
%      std::endl;
%
%    gLog           << ss.str ();
%    fLilypondCodeStream << ss.str ();
%  }
%#endif // MF_TRACE_IS_ENABLED
%
%
```

Chapter 20

Music Scores Representation (MSR)

MSR is the central format of music scores in MusicFormats. It contains a very detailed representation of western notation music score elements. Most of it is handling music in a sequential way. See chapter 21 [MSR time-oriented represention], page 218, for a presentation of how it handles time-oriented concerns.

Some of the data in MSR are supplied by the code that uses MSR, as in class `msrSlur`:

```
    static SMARTP<msrSlur> create (
                            int               inputLineNumber,
                            int               slurNumber,
                            msrSlurTypeKind   slurTypeKind,
                            msrLineTypeKind   slurLineTypeKind,
                            msrPlacementKind  slurPlacementKind);

    // ... ... ...

    // private fields
    // ------------------------------------------------------


    int                       fSlurNumber;

    msrSlurTypeKind           fSlurTypeKind;

    msrLineTypeKind           fSlurLineTypeKind;

    msrPlacementKind          fSlurPlacementKind;
```

Other data are computed by the MSR private methods. For example, in `src/formats/msr/msrVoices.h`:

```
    // there can only be 4 regular voices in a staff
    // (those that can contain beamed notes)
    // and we need a number for the orientation of beams
    int                       fRegularVoiceStaffSequentialNumber;

    // ... ... ...

    // fVoiceShortestNoteWholeNotes and fVoiceShortestNoteTupletFactor
    // are used to compute a number of divisions per quarter note
    // if needed, such as when generating MusicXML from MSR
    mfRational                  fVoiceShortestNoteWholeNotes;
    msrTupletFactor           fVoiceShortestNoteTupletFactor;
```

There are also data that varies during the lifetime of the object, while it is being populated for example. One such case is class `msrMeasure`:

```
    mfRational                fCurrentMeasureCurrentAccumulatedWholeNotesDuration;
                              // this increases when musical elements
                              // are appended to the measure
```

MSR has been designed to be as general as possible, leading it to contain informations fitted to the various textual formats that can be converted to it or output from it by MusicFormats services.

It is a *very fine-grained* representation of scores:

- some informations it contains are present as such in the textual formats;
- others are computed when the representation is populated, such as, in src/formats/msr/msrVoices.h:

```
    mfRational                fVoiceShortestNoteWholeNotes;
```

This information is used when generating MusicXML output to set the `<divisions/>` value.

LPSR and BSR contain an MSR as a sub-component, in order to allow for easy two-way conversion. This avoids the loss of information. This is why converting LPSR and BSR to MSR is done at no cost: just get the MSR component.

Both LPSR and BSR complement their MSR sub-component with whatever is needed for their purpose:

- LPSR contains a description of the structure of the score for the needs of LilyPond output and export from LilyPond when this becomes available;
- BSR contains a description of how to layout the braille cell on the embossed page, in terms of cells per line and lines per page.

20.1 MSR basic types

Some types used thoughout MSR are defined in src/formats/msr/msrBasicTypes.h/.cpp:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msr >  egrep -rIn  '^// '
    msrBasicTypes.h
msrBasicTypes.h:29:// input line numbers
msrBasicTypes.h:34:// names lists max length
msrBasicTypes.h:35:// ------------------------------------------------------
msrBasicTypes.h:39:// XMLLang
msrBasicTypes.h:52:// diatonic pitches
msrBasicTypes.h:69:// alterations
msrBasicTypes.h:90:// accidentals
msrBasicTypes.h:124:// editorial accidentals
... ... ...
msrBasicTypes.h:1840:// moments
msrBasicTypes.h:1938:// tuplet factors
msrBasicTypes.h:2024:// harmonies intervals
msrBasicTypes.h:2134:// harmonies structure
msrBasicTypes.h:2231:// harmonies contents
msrBasicTypes.h:2320:// harmonies details and analysis
msrBasicTypes.h:2333:// RGB colors
msrBasicTypes.h:2391:// AlphaRGB colors
msrBasicTypes.h:2444:// score notation kinds
msrBasicTypes.h:2455:// global variables
msrBasicTypes.h:2500:// initialization
```

20.2 Data matching across formats

Choices have to be made regarding the way we represent music scores elements, since this varies across formats.

In particular, the way MusicXML structures the elements is not what MSR does. For example, class `msrIdentification` in `src/formats/msr/msrIdentification.h` contains:

```
class EXP msrIdentification : public msrElement
{
 // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // work
    std::string           fWorkNumber;

    // ... ... ...

    // creators

    // ... ... ...

    std::list<std::string>
                          fSoftwaresList;

  // ... ... ...
};
```

This information is stored in distinct elements in MusicXML:

```
<score-partwise>
  <work>
    <work-number>K. 331</work-number>
    <work-title>Piano Sonata in A Major</work-title>
  </work>
  <identification>
    <creator type="composer">Wolfgang Amadeus Mozart</creator>
    <rights>Copyright © 2003 Recordare LLC</rights>
    <encoding>
      <software>Finale 2003 for Windows</software>
      <software>Dolet for Finale 1.3</software>
      <encoding-date>2003-03-14</encoding-date>
    </encoding>
```

The same occurs for MusicXML's `<direction/>` elements, that contain distinct subelements `<words/>` and `<metronome/>`:

```
      <direction>
        <direction-type>
          <words>Adagio</words>
        </direction-type>
        <direction-type>
          <metronome>
            <beat-unit>long</beat-unit>
            <per-minute>100</per-minute>
          </metronome>
        </direction-type>
      </direction>
```

Note that two `<direction-type/>` elements are needed, since only one of `<words/>` and `<metronome/>` can be present in a given instance, as stated in direction.mod:

```
<!ELEMENT direction-type (rehearsalMark+ | segno+ | coda+ |
  (words | symbol)+ | wedge | dynamics+ | dashes |
  bracket | pedal | metronome | octave-shift | harp-pedals |
  damp | damp-all | eyeglasses | std::string-mute |
  scordatura | image | principal-voice | percussion+ |
  accordion-registration | staff-divide | other-direction)>
```

This is not a problem in GUI applications, since all those elements are simply *drawn*. MSR stores this in a single class `msrTempo` class in src/formats/msr/msrTempos.h/.cpp, since musicians use tempo indications as a whole. See chapter 42 [Tempos handling], page 313 and section 20.17 [Tempos], page 193 for more details.

20.3 Lengths

There are several cases where a length is used in MSR, hence:

```
enum class msrLengthUnitKind {
  kUnitInch, kUnitCentimeter, kUnitMillimeter
};
```

```
class EXP msrLength : public smartable
{
    // ... ... ...

    msrLengthUnitKind     fLengthUnitKind;
    float                 fLengthValue;
```

20.4 Sounding and displayed durations

All durations are represented by mfRational numbers whose denominators are powers of 2, such as `mfRational (3, 16`, and relative to the duration of a whole note.

This information is a field of class `msrMeasureElement`:

```
    mfRational              fMeasureElementSoundingWholeNotes;
```

In a tuplet, the sounding durations are different than the written durations, so we store the written duration in class `msrNote`:

```
    // whole notes
    mfRational              fNoteDisplayWholeNotes;

    int                   fNoteDotsNumber;

    msrNotesDurationKind  fNoteGraphicNotesDurationKind;

    msrTupletFactor       fNoteTupletFactor;

    msrQuarterTonesPitchKind
                          fNoteQuarterTonesDisplayPitchKind;
    msrOctaveKind         fNoteDisplayOctaveKind;
                                // for unpitched notes
                                // and pitched rests
```

Enumeration type `msrNotesDurationKind` is declared in src/formats/msr/msrNotesDurations.h:

```
enum class msrNotesDurationKind {
  kNotesDuration_UNKNOWN_,

  // from longest to shortest for the algorithms
  kNotesDurationMaxima, kNotesDurationLonga, kNotesDurationBreve,
  kNotesDurationWhole, kNotesDurationHalf,
  kNotesDurationQuarter,
  kNotesDurationEighth, kNotesDuration16th, kNotesDuration32nd, kNotesDuration64th,
  kNotesDuration128th, kNotesDuration256th, kNotesDuration512th, kNotesDuration1024th
};
```

20.5 Measure positions and moments

Measure positions are represented by `mfRational` numbers such as `3/8`, `1/1` being a whole note.

Measure positions are stored in field `msrMeasureElement::fMeasureElementMeasurePosition` in class `msrMeasureElement`, defined in `src/formats/msr/msrMeasureElement.h/.cpp`:

```
class EXP msrMeasureElement : public msrElement
{
  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    /*
      The uplink to measure is declared in the sub-classes,
      to allow for separate *.h files, C++ constraint
    */

    mfRational               fMeasureElementSoundingWholeNotes;

    std::string            fBarLineUpLinkToMeasure->getMeasureNumber ();

    mfRational               fMeasureElementMeasurePosition;
    mfRational               fMeasureElementVoicePosition;
};
```

LilyPond represents grace notes positions with a so-called *moment*, that complement the measure position with a relative offset. Grace notes durations are not accounted for in the whole notes duration of measures.

Class `msrMoment` stores a position in a measure, with a relative offset since harmonies can be placed on a note during its sounding time:

```
    mfRational               fWrittenPositionInMeseasure;
    mfRational               fSoundingRelativeOffset;
```

20.6 Rests and skips

A skip is an invisible rest, i.e. the meaning is the same as that in LilyPond. Skips are created to it fill the holes between notes wherever needed, in order for all voices to be notes/rests/skips sequences.

Skips are also created in `src/passes/msr2lpsr/` to circumvent the LilyPond #34 issue.

20.7 Solo notes and rests

A solo note or rest is characterized as sounding alone in its multi-voice staff for its whole duration.

In the case of a solo rests, such detection allows for better output, in particular when LilyPond code is generated.
An example is at figure 20.1 [The solo rests problem], page 186 : the eighth rests in the second measure of voice 1 of the first staff should be be placed on the middle line of the staff, as `MuseScore` does.

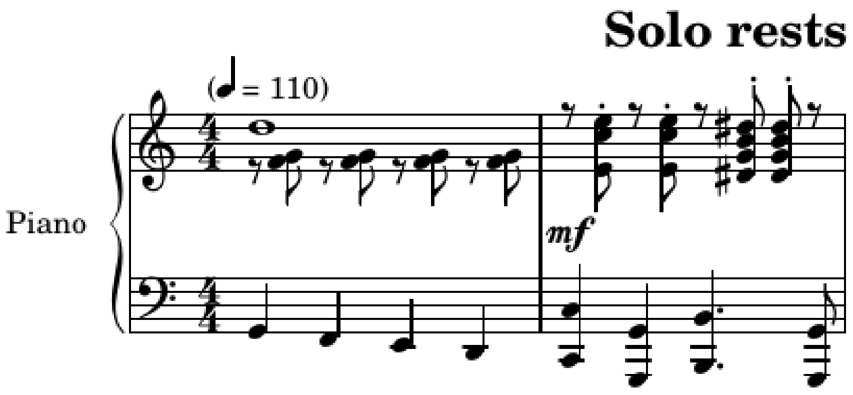


Figure 20.1: The solo rests problem

20.8 Linear versus time-oriented representation

Most music scoring GUI applications handle music as containing voices, which are made of sequences of notes, chords, tuplets and such. This is a horizontal, linear view of the music in the score.

Another view of the music is time-oriented, i.e., what are are notes being played at a given moment in time? This is a vertical view of the music, which is highlighted in piano roll views.

MSR stores descriptions of so-called 'measures slice' through class `msrMeasuresSlice`, defined in `src/formats/msr/ms`
Then a time-oriented view of a voice, staff or part is a sequence of such measure slices, defined in class `msrMeasuresSlicesSequence`.

An class `msrMeasuresSlice` contains basically a slice measures vector:

```
    // the measures in the slice
    std::vector<S_msrMeasure>  fSliceMeasuresVector;
```

From this, the following other descriptions are derived:

```
    // notes flat list
    std::list<S_msrNote>  fSliceNotesFlatList;

    // note events list
    std::list<S_msrNoteEvent>
                          fSliceNoteEventsList;

    // simultaneous notes chunks list
    std::list<S_msrSimultaneousNotesChunk>
                          fSliceSimultaneousNotesChunksList;
```

Note events are distinguished with enumeration type `msrNoteEventKind`:

```
//______________________________________________________________________
enum class msrNoteEventKind {
  kNoteEventStart,
  kNoteEventStop
};
```

Class `msrNoteEvent` contains:

```
    mfRational                fNoteEventMeasurePosition;
    S_msrNote               fNoteEventNote;
    msrNoteEventKind        fNoteEventKind;
```

20.9 Spanners

A spanner... spans from one note or rest to another one. A choice to be made about when to use spanners: should wedges `<` and `>` be handled as spanners, or simply as being attached to notes? It has been selected to use spanners only for ligatures apart from true spanners.

MusicXML uses `"start"`, `"start"` and `"start"` attributes, which need to be present in MSR for MusicXML generation. They are reflected in MSR as enumeration type enumeration type `msrSpannerTypeKind`, defined this way:

```
// spanner types
//______________________________________________________________________________
enum class msrSpannerTypeKind {
  kSpannerType_UNKNOWN_,

  kSpannerTypeStart, kSpannerTypeContinue, kSpannerTypeStop
};
```

20.10 Uplinks, direct uplinks and sidelinks

An uplink is a direct pointer from one class instance to one that contains it. Some are are a link to the containing class instance, whilst others are shortcut links higher in the graph for speed. For example, class `msrNote` contains:

```
    // upLinks
    // ------------------------------------------------------

    S_msrMeasure            fNoteUpLinkToMeasure;

    S_msrChord              fNoteShortcutUpLinkToChord;

    S_msrGraceNotesGroup    fNoteShortcutUpLinkToGraceNotesGroup;

    S_msrTuplet             fNoteShortcutUpLinkToTuplet;
```

A sidelink is used in ligatures and spanners, so that each end of the structure can reference the other one.

For example, MusicFormats defines enumeration type `msrLigatureKind` in `src/formats/msr/msrLigatures.h`:

```
enum class msrLigatureKind {
  kLigatureNone,
  kLigatureStart, kLigatureContinue, kLigatureStop
};
```

Class `msrLigature` contains:

```
  private:

    // sideLinks
    // ------------------------------------------------------
    S_msrLigature           fLigatureSideLinkToOtherEnd; // two-way
```

Enumeration type is declared in `src/formats/msr/msrSpanners.h`:

```
enum class msrSpannerKind {
  kSpannerDashes, kSpannerWavyLine
};
```

20.11 Printing descriptions

There is a standard set of methods to print the contents of the descriptions in MusicFormats to standard output, depending on the granularity of the information to be displayed:

```
    void                      print (std::ostream& os) const override;

    std::string               asString () const override;
    std::string               asStringShort () const override;
```

There are also more specific methods such as:

```
    void                      printFull (std::ostream& os) const override;

    void                      printSummary (std::ostream& os) const override;
```

Note that:

- virtual method `asString ()` produces a rather condensed view of the data to be displayed as part of a single line;
- virtual method `print ()` may produce its output on multiples lines, which always ends with an end of line.

Most classes in MusicFormats can be printed with the `<<` operator:

```
std::ostream& operator << (std::ostream& os, const S_msrElement& elt)
{
  if (elt) {
    elt->print (os);
  }
  else {
    os << "[NULL]" << std::endl;
  }

  return os;
}
```

In simple cases, virtual method `print ()` merely calls virtual method `asString ()`:

```
void msrElement::print (std::ostream& os) const
{
  os << asString () << std::endl;
}
```

All virtual method `asString ()` methods produce an output of the form `[...]`, in order to facilitate selecting the whole with a double click to help the user, since such output can be nested:

```
std::string msrTransposition::asString () const
{
  std::stringstream ss;

  ss <<
    "[Transposition" <<
    ", fTranspositionDiatonic = " << fTranspositionDiatonic <<
    ", fTranspositionChromatic = " << fTranspositionChromatic <<
    ", fTranspositionOctaveChange = " << fTranspositionOctaveChange <<
    ", fTranspositionDouble = " << fTranspositionDouble <<
    ", line " << fInputStartLineNumber <<
    ']';

  return ss.str ();
}
```

A typical sequence to produce indented output is:

```
void msrTransposition::print (std::ostream& os) const
{
  const int fieldWidth = 22;

  os <<
    "Transposition" <<
    ", line " << fInputStartLineNumber <<
    std::endl;

  ++gIndenter;

  os << std::left <<
    std::setw (fieldWidth) <<
    "fTranspositionDiatonic" << ": " << fTranspositionDiatonic <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionChromatic" << ": " << fTranspositionChromatic <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionOctaveChange" << ": " << fTranspositionOctaveChange <<
    std::endl <<
    std::setw (fieldWidth) <<
    "fTranspositionDouble" << ": " << fTranspositionDouble <<
    std::endl << std::endl;

  --gIndenter;
}
```

The main indented output streams are:

```
#define gOutput *gGlobalOutputIndentedOstream
#define gLog    *gGlobalLogIndentedOstream
```

20.12 Pitches

MSR handle diatonic, semitone and quarter tone pitches, defined in `src/formats/msr/msrBasicTypes.h` as shown below. All pitches data is represented internally as quater tones pitches, and conversions are done wherever needed.

```
// diatonic pitches
//______________________________________________________________________________
enum class msrDiatonicPitchKind {
  kDiatonicPitch_UNKNOWN_,

  // starting at C for LilyPond relative octave calculations
  kDiatonicPitchC,
  kDiatonicPitchD, kDiatonicPitchE, kDiatonicPitchF,
  kDiatonicPitchG, kDiatonicPitchA, kDiatonicPitchB
};
```

```
// semi tones pitches
//______________________________________________________________________________
enum class msrSemiTonesPitchKind {
  kSTP_NoSemiTonesPitch,

  kSTP_C_TripleFlat,
  kSTP_C_DoubleFlat, kSTP_C_Flat,
  kSTP_C_Natural,
  kSTP_C_Sharp, kSTP_C_DoubleSharp,
  kSTP_C_TripleSharp,

  kSTP_D_TripleFlat,
  kSTP_D_DoubleFlat, kSTP_D_Flat,
  kSTP_D_Natural,
  kSTP_D_Sharp, kSTP_D_DoubleSharp,
  kSTP_D_TripleSharp,

  kSTP_E_TripleFlat,
  kSTP_E_DoubleFlat, kSTP_E_Flat,
  kSTP_E_Natural,
  kSTP_E_Sharp, kSTP_E_DoubleSharp,
  kSTP_E_TripleSharp,

  kSTP_F_TripleFlat,
  kSTP_F_DoubleleFlat, kSTP_F_Flat,
  kSTP_F_Natural,
  kSTP_F_Sharp, kSTP_F_DoubleSharp,
  kSTP_F_TripleSharp,

  kSTP_G_TripleFlat,
  kSTP_G_DoubleFlat, kSTP_G_Flat,
  kSTP_G_Natural,
  kSTP_G_Sharp, kSTP_G_DoubleSharp,
  kSTP_G_TripleSharp,

  kSTP_A_TripleFlat,
  kSTP_A_DoubleFlat, kSTP_A_Flat,
  kSTP_A_Natural,
  kSTP_A_Sharp, kSTP_A_DoubleSharp,
  kSTP_A_TripleSharp,

  kSTP_B_TripleFlat,
  kSTP_B_DoubleFlat, kSTP_B_Flat,
  kSTP_B_Natural,
  kSTP_B_Sharp, kSTP_B_DoubleSharp,
  kSTP_B_TripleSharp
};
```

```
// quarter tones pitches
//______________________________________________________________________________
enum class msrQuarterTonesPitchKind {
  kQTP_UNKNOWN_,

  kQTP_Rest, kQTP_Skip,

  kQTP_A_TripleFlat,
  kQTP_A_DoubleFlat, kQTP_A_SesquiFlat, kQTP_A_Flat, kQTP_A_SemiFlat,
  kQTP_A_Natural,
  kQTP_A_SemiSharp, kQTP_A_Sharp, kQTP_A_SesquiSharp, kQTP_A_DoubleSharp,
  kQTP_A_TripleSharp,

  kQTP_B_TripleFlat,
  kQTP_B_DoubleFlat, kQTP_B_SesquiFlat, kQTP_B_Flat, kQTP_B_SemiFlat,
  kQTP_B_Natural,
  kQTP_B_SemiSharp, kQTP_B_Sharp, kQTP_B_SesquiSharp, kQTP_B_DoubleSharp,
  kQTP_B_TripleSharp,

  kQTP_C_TripleFlat,
  kQTP_C_DoubleFlat, kQTP_C_SesquiFlat, kQTP_C_Flat, kQTP_C_SemiFlat,
  kQTP_C_Natural,
  kQTP_C_SemiSharp, kQTP_C_Sharp, kQTP_C_SesquiSharp, kQTP_C_DoubleSharp,
  kQTP_C_TripleSharp,

  kQTP_D_TripleFlat,
  kQTP_D_DoubleFlat, kQTP_D_SesquiFlat, kQTP_D_Flat, kQTP_D_SemiFlat,
  kQTP_D_Natural,
  kQTP_D_SemiSharp, kQTP_D_Sharp, kQTP_D_SesquiSharp, kQTP_D_DoubleSharp,
  kQTP_D_TripleSharp,

  kQTP_E_TripleFlat,
  kQTP_E_DoubleFlat, kQTP_E_SesquiFlat, kQTP_E_Flat, kQTP_E_SemiFlat,
  kQTP_E_Natural,
  kQTP_E_SemiSharp, kQTP_E_Sharp, kQTP_E_SesquiSharp, kQTP_E_DoubleSharp,
  kQTP_E_TripleSharp,

  kQTP_F_TripleFlat,
  kQTP_F_DoubleFlat, kQTP_F_SesquiFlat, kQTP_F_Flat, kQTP_F_SemiFlat,
  kQTP_F_Natural,
  kQTP_F_SemiSharp, kQTP_F_Sharp, kQTP_F_SesquiSharp, kQTP_F_DoubleSharp,
  kQTP_F_TripleSharp,

  kQTP_G_TripleFlat,
  kQTP_G_DoubleFlat, kQTP_G_SesquiFlat, kQTP_G_Flat, kQTP_G_SemiFlat,
  kQTP_G_Natural,
  kQTP_G_SemiSharp, kQTP_G_Sharp, kQTP_G_SesquiSharp, kQTP_G_DoubleSharp,
  kQTP_G_TripleSharp
};
```

20.13 Octaves

They are represented with enumeration type :

```
// octaves
//______________________________________________________________________________
enum class msrOctaveKind {
  kOctave_UNKNOWN_,

  kOctave0, kOctave1, kOctave2, kOctave3,
  kOctave4, // that of middle C
  kOctave5, kOctave6, kOctave7, kOctave8, kOctave9
};
```

For the needs of LilyPond and MSDL, MSR also contains a description of how to enter octaves:

```
// octave entry
//______________________________________________________________________________
enum class msrOctaveEntryKind {
  kOctaveEntryRelative,
  kOctaveEntryAbsolute,
  kOctaveEntryFixed
};
```

20.14 NotesDurations

They are represented in MSR with the enumeration type `msrNotesDurationKind` enumeration type, defined in `src/formats/msr/msrBasicTypes.h`:

```
// durations
//______________________________________________________________________________
enum class msrNotesDurationKind {
  kNotesDuration_UNKNOWN_,

  // from longest to shortest for the algorithms
  kNotesDurationMaxima, kNotesDurationLonga, kNotesDurationBreve, kNotesDurationWhole,
    kNotesDurationHalf,
  kNotesDurationQuarter,
  kNotesDurationEighth, kNotesDuration16th, kNotesDuration32nd, kNotesDuration64th,
    kNotesDuration128th, kNotesDuration256th, kNotesDuration512th, kNotesDuration1024th
};
```

Class `msrDottedNotesDuration` contains:

```
    msrNotesDurationKind  fNotesDurationKind;
    int                   fDotsNumber;
```

20.15 Alterations

```
// alterations
//______________________________________________________________________________
enum class msrAlterationKind {
  kAlteration_UNKNOWN_,

  kAlterationTripleFlat, kAlterationDoubleFlat, kAlterationSesquiFlat,
  kAlterationFlat, kAlterationSemiFlat,
  kAlterationNatural,
  kAlterationSemiSharp, kAlterationSharp, kAlterationSesquiSharp,
  kAlterationDoubleSharp, kAlterationTripleSharp
```

20.16 Accidentals

```
// accidentals
//______________________________________________________________________________
enum class msrAccidentalKind {
  kAccidentalNone,

  kAccidentalSharp, kAccidentalNatural,
  kAccidentalFlat, kAccidentalDoubleSharp,
  kAccidentalSharpSharp,
  kAccidentalFlatFlat, kAccidentalNaturalSharp,
  kAccidentalNaturalFlat, kAccidentalQuarterFlat,
  kAccidentalQuarterSharp,kAccidentalThreeQuartersFlat,
  kAccidentalThreeQuartersSharp,

  kAccidentalSharpDown, kAccidentalSharpUp,
  kAccidentalNaturalDown, kAccidentalNaturalUp,
  kAccidentalFlatDown, kAccidentalFlatUp,
  kAccidentalTripleSharp, kAccidentalTripleFlat,
  kAccidentalSlashQuarterSharp, kAccidentalSlashSharp,
  kAccidentalSlashFlat, kAccidentalDoubleSlashFlat,
  kAccidentalSharp_1, kAccidentalSharp_2,
  kAccidentalSharp_3,  kAccidentalSharp_5,
  kAccidentalFlat_1, kAccidentalFlat_2,
  kAccidentalFlat_3, kAccidentalFlat_4,
  kAccidentalSori, kAccidentalKoron,

  kAccidentalOther
};
```

20.17 Tempos

There are thus several kinds of tempos in MSR, with variants represented by enumeration type `msrTempoKBeatUnitsKi` in `src/formats/msr/msrTempos.h`:

```
class EXP msrTempo : public msrMeasureElement
{
  public:

    // data types
    // ------------------------------------------------------

    enum class msrTempoKBeatUnitsKind {
      kTempoBeatUnits_UNKNOWN_,
      kTempoBeatUnitsWordsOnly,
      kTempoBeatUnitsPerMinute,
      kTempoBeatUnitsEquivalence,
      kTempoNotesRelationship
    };

    // ... ... ...

    enum class msrTempoParenthesizedKind {
      kTempoParenthesizedYes, kTempoParenthesizedNo
    };

    // ... ... ...

    enum class msrTempoNotesRelationshipKind {
      kTempoNotesRelationshipNone, kTempoNotesRelationshipEquals
    };

    // ... ... ...
};
```

20.17.1 Tempos notes

A tempo indication can contain a note a notes in a tuplet. Such notes are described by class `msrTempoNote`:

```
class EXP msrTempoNote : public msrElement
{
  public:

    // creation from MusicXML
    // ------------------------------------------------------

    static SMARTP<msrTempoNote> create (
                                int               inputLineNumber,
                                const mfRational& tempoNoteWholeNotes,
                                Bool              tempoNoteBelongsToATuplet);

  protected:

    // constructors/destructor
    // ------------------------------------------------------

                              msrTempoNote (
                                int               inputLineNumber,
                                const mfRational& tempoNoteWholeNotes,
                                Bool              tempoNoteBelongsToATuplet);

    // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    mfRational                fTempoNoteWholeNotes;

    std::list<S_msrBeam>    fTempoNoteBeamsList;

    Bool                    fTempoNoteBelongsToATuplet;
};
```

20.17.2 Tempos tuplets

A tuplet in a tempo representation is described by class `msrTempoTuplet`:

```
//______________________________________________________________________________
class EXP msrTempoTuplet : public msrElement
{
  public:

    // data types
    // ------------------------------------------------------

    enum class msrTempoTupletTypeKind {
      kTempoTupletTypeNone,
      kTempoTupletTypeStart, kTempoTupletTypeStop
    };

    // ... ... ...

    enum class msrTempoTupletBracketKind {
      kTempoTupletBracketYes, kTempoTupletBracketNo
    };

    // ... ... ...

    enum class msrTempoTupletShowNumberKind {
      kTempoTupletShowNumberActual,
      kTempoTupletShowNumberBoth,
      kTempoTupletShowNumberNone
    };

    // ... ... ...

    // creation from MusicXML
    // ------------------------------------------------------

    static SMARTP<msrTempoTuplet> create (
      int                             inputLineNumber,
      int                             tempoTupletNumber,
      msrTempoTupletBracketKind       tempoTupletBracketKind,
      msrTempoTupletShowNumberKind tempoTupletShowNumberKind,
      msrTupletFactor                 tempoTupletFactor,
      msrWholeNotes               memberNotesDisplayWholeNotes);

  protected:

    // constructors/destructor
    // ------------------------------------------------------

    msrTempoTuplet (
      int                             inputLineNumber,
      int                             tempoTupletNumber,
      msrTempoTupletBracketKind       tempoTupletBracketKind,
      msrTempoTupletShowNumberKind tempoTupletShowNumberKind,
      msrTupletFactor                 tempoTupletFactor,
      msrWholeNotes               memberNotesDisplayWholeNotes);

    // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    int                       fTempoTupletNumber;

    msrTempoTupletBracketKind
                              fTempoTupletBracketKind;

    msrTempoTupletShowNumberKind
                              fTempoTupletShowNumberKind;

    msrTupletFactor           fTempoTupletFactor;

    mfRational                  fMemberNotesDisplayWholeNotes;

    mfRational                  fTempoTupletDisplayWholeNotes;

    std::list<S_msrElement>
                              fTempoTupletElementsList;
};
```

20.17.3 Tempos description

The private fields in class `msrTempo` are:

```
class EXP msrTempo : public msrMeasureElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    msrTempoKBeatUnitsKind
                              fTempoKind;

    std::list<S_msrWords> fTempoWordsList;

    msrDottedNotesDuration     fTempoBeatUnit;

    std::string            fTempoPerMinute; // '90' or '132-156' for example
    msrDottedNotesDuration     fTempoEquivalentBeatUnit;

    S_msrTempoNotesRelationshipElements
                           fTempoNotesRelationshipLeftElements;
    msrTempoNotesRelationshipKind
                           fTempoNotesRelationshipKind;
    S_msrTempoNotesRelationshipElements
                           fTempoNotesRelationshipRightElements;

    msrTempoParenthesizedKind
                           fTempoParenthesizedKind;

    msrPlacementKind       fTempoPlacementKind;
};
```

Among these fields:

- field `msrTempo::fTempoKind` denotes the variant;
- field `msrTempo::fTempoWordsList` contains the words that can be present, such as 'adagio molto';
- field `msrTempo::fTempoBeatUnit` is a dotted duration, as in '4.';
- field `msrTempo::fTempoPerMinute` is a std::string, since it can contain ranges indication as in '4. = 60-66';
- field `msrTempo::fTempoEquivalentBeatUnit` is a dotted duration;
- field `msrTempo::fTempoNotesRelationshipLeftElements`, field `msrTempo::fTempoNotesRelationshipKind` and field `msrTempo::fTempoNotesRelationshipRightElements` are used when a relationship is present, such as '2. = 1', in which case field `msrTempo::fTempoNotesRelationshipKind` contains field `msrTempo::kTemp`
- field `msrTempo::fTempoParenthesizedKind` indicates whether the tempo indication is parenthesized;
- field `msrTempo::fTempoPlacementKind` tells whether the tempo is to be places above or below the staff, constant `msrPlacementKind::kPlacementAbove` by default.

20.18 Clefs

Clefs are distinguished using enumeration type `msrClefKind`:

```
// clefs
//______________________________________________________________________________

enum class msrClefKind {
  kClef_UNKNOWN_,

  kClefTreble,
  kClefSoprano, kClefMezzoSoprano, kClefAlto, kClefTenor, kClefBaritone, kClefBass,
  kClefTrebleLine1,
  kClefTrebleMinus15, kClefTrebleMinus8, kClefTreblePlus8, kClefTreblePlus15,

  kClefBassMinus15, kClefBassMinus8, kClefBassPlus8, kClefBassPlus15,

  kClefVarbaritone,

  kClefTablature4, kClefTablature5, kClefTablature6, kClefTablature7,

  kClefPercussion,

  kClefJianpu
};
```

Class `msrClef` contains:

```
    msrClefKind             fClefKind;
    int                     fClefStaffNumber;
```

20.19 Keys

MSR, as MusicXML, suports Humdrum-Scot keys as well as traditional key such as `C` and `6/8`.

A Humdrum-Scot key is composed of items represented by class `msrHumdrumScotKeyItem`, each containing:

```
    msrDiatonicPitchKind  fKeyDiatonicPitchKind;
    msrAlterationKind     fKeyAlterationKind;
    msrOctaveKind         fKeyOctaveKind;
```

An example is at figure 20.2 [Humdrum-Scot keys], page 197. It has been produced by:

```
xml2ly -auto-output-file-name keys/HumdrumScotKeys.xml
```

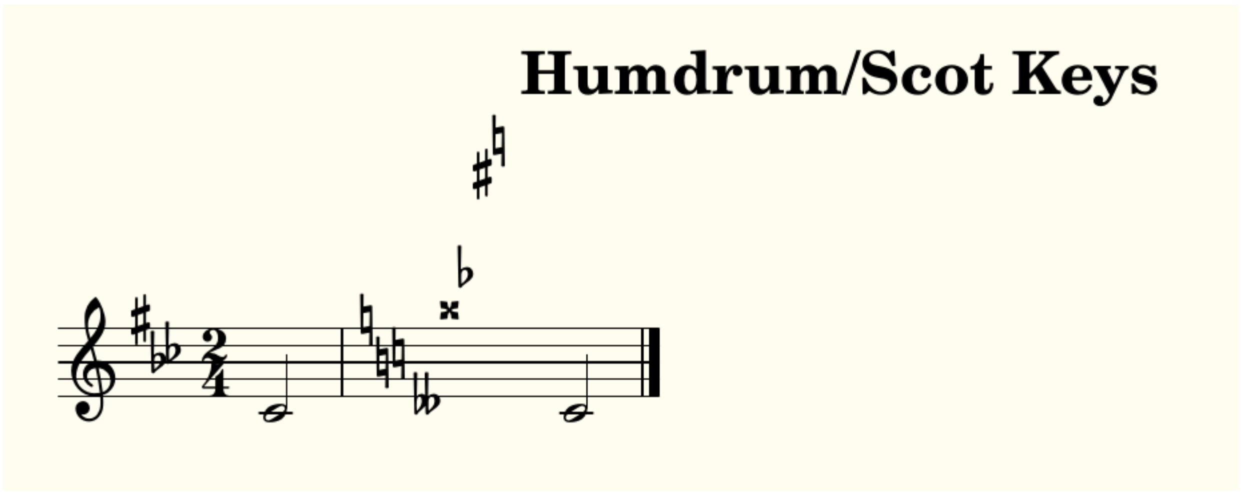


Figure 20.2: Humdrum-Scot keys

Class `msrKey` thus contains:

```
    // private fields
    // ------------------------------------------------------

    msrKeyKind              fKeyKind;

    // traditional keys

    msrQuarterTonesPitchKind
                            fKeyTonicQuarterTonesPitchKind;
    msrModeKind             fModeKind;
    int                     fKeyCancel;

    // Humdrum/Scot keys

    std::vector<S_msrHumdrumScotKeyItem>
```

```
                            fHumdrumScotKeyItemsVector;
    Bool                    fKeyItemsOctavesAreSpecified;
```

20.20 Time signatures

The variants in time signatures are distinguished by enumeration type enumeration type `msrTimeSignatureSymbolKi`

```
// time symbols
//______________________________________________________________________________
enum class msrTimeSignatureSymbolKind {
  kTimeSignatureSymbolNone,
  kTimeSignatureSymbolCommon,
  kTimeSignatureSymbolCut,
  kTimeSignatureSymbolNote,
  kTimeSignatureSymbolDottedNote,
  kTimeSignatureSymbolSingleNumber,
  kTimeSignatureSymbolSenzaMisura
};
```

A time signature can also be structured, and this is described by those two types:

```
enum class msrTimeSignatureSeparatorKind {
  kTimeSignatureSeparatorNone,
  kTimeSignatureSeparatorHorizontal,
  kTimeSignatureSeparatorDiagonal,
  kTimeSignatureSeparatorVertical,
  kTimeSignatureSeparatorAdjacent
};
```

```
enum class msrTimeSignatureRelationKind {
  kTimeSignatureRelationNone,
  kTimeSignatureRelationParentheses,
  kTimeSignatureRelationBracket,
  kTimeSignatureRelationEquals,
  kTimeSignatureRelationSlash,
  kTimeSignatureRelationSpace,
  kTimeSignatureRelationHyphen
};
```

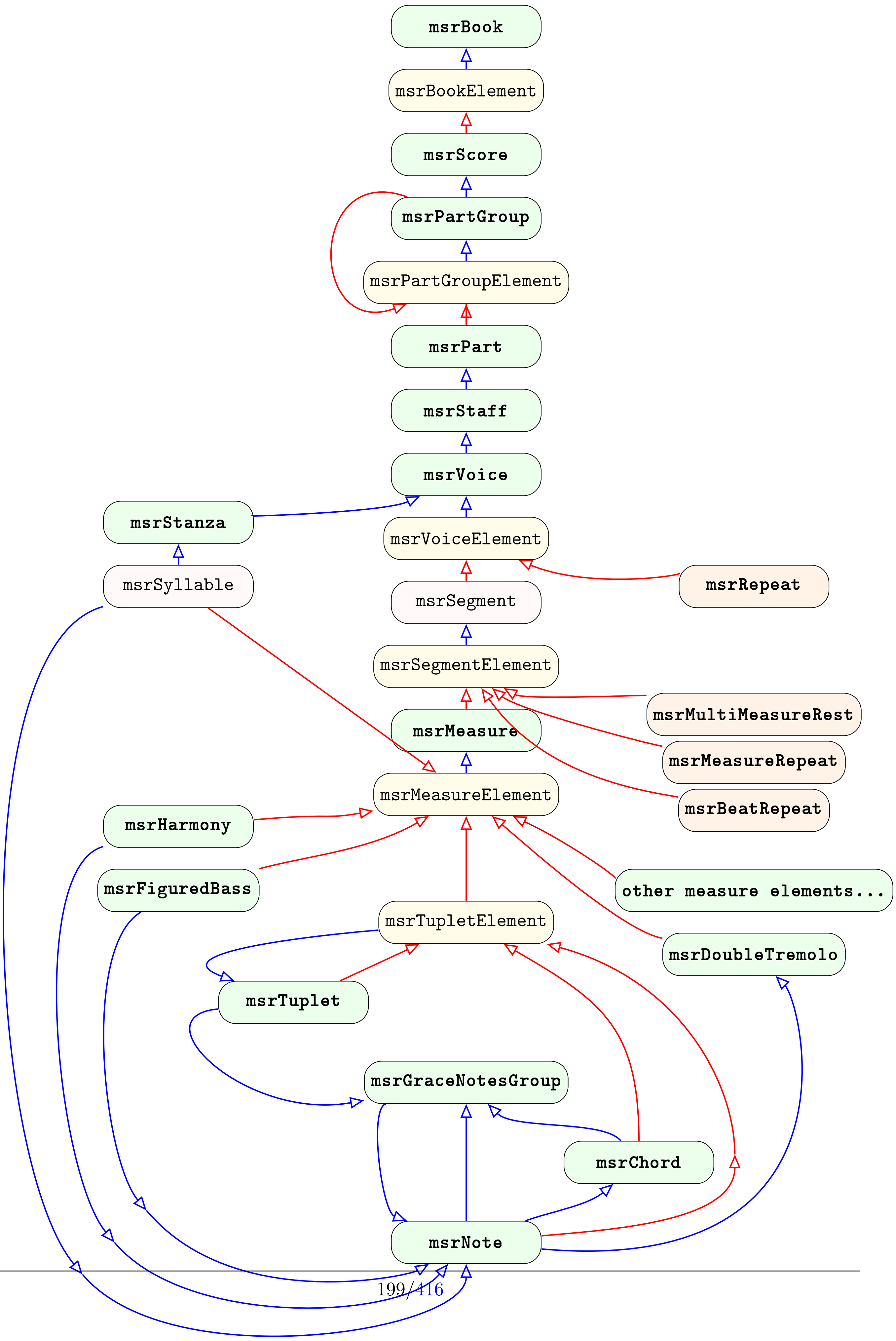
A brick that can be used in class `msrTimeSignature` is `msrTimeSignatureItem`, whose private fields are:

```
    std::vector<int>        fTimeSignatureBeatsNumbersVector; // 5+3+1 is possible
    int                     fTimeSignatureBeatValue;
```

Class `msrTimeSignature` contains:

```
    msrTimeSignatureSymbolKind
                            fTimeSignatureSymbolKind;

    std::vector<S_msrTimeSignatureItem>
                            fTimeSignatureItemsVector;

    // a time is compound if it contains several items
    // or if the only one has several beats numbers
    // i.e. 3/4 is not, (3+4)/8 is, and 2/4+3/4 is too
    Bool                    fTimeIsCompound;
```

Figure 20.3: The MSR classes hierarchy



20.21 MSR classes inheritance

The picture at figure 20.3 [The MSR classes hierarchy], page 199, shows the hierarchy of the main MSR classes. The colors are used as follows:

The background colors are used as follows:

- green: a score element that is expected to be found in a score representation, such as class `msrStaff` and class `msrChord`;
- pink: a element needed in MSR to structure the representation, such as class `msrSegment` and class `msrSyllable`;
- yellow: a base class with name class `msr*Element` for elements that can be used in another class, such as class `msrVoiceElement`;

The arrows colors have the following meaning:

- red: a link from a class to its base class. For example, class `msrPart` is derived from class `msrPartGroupElement`, class `msrPartGroup` is derived from class `msrPartGroupElement`, and class `msrChord` is derived from class `msrTupletElement`;
- blue: one or more fields of a class are smart pointers to instances of another. For example, an class `msrChords` instance may be an element of a class `msrGraceNotesGroup` instance.

When not shown for clarity, the common base class of all these classes is class `msrElement`, that contains an integer input line number.

The `otherMeasureElements` classes are:

- bars:
  - class `msrBarCheck`
  - class `msrBarNumberCheck`
  - class `msrBarLine`
  - class `msrHiddenMeasureAndBarLine`
- breaks:
  - class `msrLineBreak`
  - class `msrPageBreak`
- notes:
  - class `msrVoiceStaffChange`
  - class `msrOctaveShift`
- clefs, keys, times, tempo:
  - class `msrClef`
  - class `msrKey`
  - class `msrTime`
  - class `msrTempo`

- instruments:
  - class `msrStaffDetails`
  - class `msrScordatura`
  - class `msrAccordionRegistration`
  - class `msrHarpPedalsTuning`
  - class `msrPedal`
  - class `msrDamp`
  - class `msrDampAll`
- lyrics:
  - class `msrSyllable`
- rehearsals, segno and coda:
  - class `msrRehearsalMark`
  - class `msrSegno`
  - class `msrDalSegno`
  - class `msrCoda`
- others:
  - class `msrPrintLayout`
  - class `msrEyeGlasses`
  - class `msrStaffLevelElement`
  - class `msrTransposition`
  - class `msrTupletElement`

20.22 Books

Books handling is presented at section 54 [Books handling], page 338.

LilyPond handles `\book {...}` by placing the scores one after the other in the resulting PDF or SVG files. It will also generate separate MIDI files if a `\markup {...}` block is used.

There is no such concept in MusicXML, but MSR uses it for completeness, creating an implicit class `msrBook` instance if needed.

An class `msrBook` contains a list and a set of `S_msrBookElement`:

```
    // book elements
    std::set<S_msrBookElement> fBookElementsSet;

    std::list<S_msrBookElement>fBookElementsList;
```

Currently, the only book element used is the class `msrScore`, but others might come, such as texts, which LilyPond allows as `\markup {...}` :

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grep -r 'public msrBook' *
formats/msr/msrScores.h:class EXP msrScore : public msrBookElement
```

20.23 Scores

Scores handling is presented at section ?? [Scores handling], page ??.

A score in MSR is the usual music score concept. It contains a set and a list of S_msrPartGroup:

```
    // part groups
    std::set<S_msrPartGroup>    fScorePartGroupsSet;

    std::list<S_msrPartGroup>   fPartGroupsList;
```

20.24 Part groups

Part groups handling is presented at section 52 [Part groups handling], page 335.

A part group in MSR contains parts or other part groups. This concept is recursive, as it is in music score: the winds part group can oboes and horns part group, for example. An implicit outer-most part group exists in MSR if the score does not contain explicit part groups.

An class msrPartGroup thus contains parts and part groups in any order, as is found in symphonic music scores:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grep -r 'public msrPartGroupElement' *
formats/msr/msrParts.h:class EXP msrPart : public msrPartGroupElement
formats/msr/msrPartGroups.h:class EXP msrPartGroup : public msrPartGroupElement
```

which are stored in a list:

```
    // allowing for both parts and nested part groups as elements
    std::list<S_msrPartGroupElement>
                          fPartGroupElementsList;
```

20.25 Parts

Parts handling is presented at section 51 [Parts handling], page 333.

A part in MSR is composed of voices, stored in:

```
    // staves

    std::map<int, S_msrStaff>
                          getPartStaveNumbersToStavesMap;
    std::list<S_msrStaff> fPartAllStavesList;

    // harmonies

    S_msrStaff            fPartHarmoniesStaff;
    S_msrVoice            fPartHarmoniesVoice;

    // figured bass

    S_msrStaff            fPartFiguredBassStaff;
    S_msrVoice            fPartFiguredBassVoice;

    // voices

    std::list<S_msrVoice> fPartAllVoicesList;
```

20.26 Staves

Staves handling is presented at section ?? [Staves handling], page ??.

A staff contains at most 4 numbered voices, stored in:

```
    // the mapping of all the voices in the staff,
    // including harmonies and figured bass voices
    std::map<int, S_msrVoice>
                          fStaffVoiceNumbersToAllVoicesMap;

    // the mapping of voice numbers to regular voices
    std::map<int, S_msrVoice>
                          fStaffVoiceNumbersToRegularVoicesMap;

    // we need to handle the regular voice specifically
    // to assign them sequencing numbers from 1 to gMaxStaffVoices,
    // needed to set the beams orientation (up or down)
    int                   fStaffRegularVoicesCounter;

    // harmonies and figured bass elements should be placed %%%JMI
    // in the first regular voice of the staff, hence:
    std::list<S_msrVoice> fStaffRegularVoicesList;

    // we need to sort the voices by increasing voice numbers,
    // but with harmonies voices right before the corresponding regular voices
    std::list<S_msrVoice> fStaffAllVoicesList;
```

20.27 Voice elements

Voices contain instances of class `msrVoiceElement`, defined in src/formats/msr/msrVoiceElements.h/.cpp:

```
//______________________________________________________________________________
/*
  Various elements can found in voices,
  hence class  msrVoiceElement
*/

class EXP msrVoiceElement : public msrElement
{
  public:

    // creation from MusicXML
    // ------------------------------------------------------

    // cloning
    // ------------------------------------------------------

  protected:

                          msrVoiceElement (
                            int inputLineNumber);

    virtual               ~msrVoiceElement ();

  /*
    The voice uplink is declared in the sub-classes,
    to allow for separate *.h files, C++ constraint
  */
};
```

The classes derived from class `msrVoiceElement` are:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msr > grep 'public msrVoiceElement
    ' *.h
msrBeatRepeats.h:class EXP msrBeatRepeat : public msrVoiceElement
msrMeasureRepeats.h:class EXP msrMeasureRepeat : public msrVoiceElement
msrRepeats.h:class EXP msrRepeat : public msrVoiceElement
msrMultiMeasureRests.h:class EXP msrMultiMeasureRest : public msrVoiceElement
msrSegments.h:class EXP msrSegment : public msrVoiceElement
```

They are describes in specific sections below.

20.28 Voices

Voices handling is presented at section 49 [Voices handling], page 331.

A voice is conceptually a sequence of `S_msrVoiceElement`, that may be:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grep -r 'public msrVoiceElement' *
formats/msr/msrMeasureRepeats.h:class EXP msrMeasureRepeat : public msrVoiceElement
formats/msr/msrRepeats.h:class EXP msrRepeat : public msrVoiceElement
formats/msr/msrMultiMeasureRests.h:class EXP msrMultiMeasureRest : public msrVoiceElement
formats/msr/msrBeatRepeats.h:class EXP msrBeatRepeat : public msrVoiceElement
formats/msr/msrSegments.h:class EXP msrSegment : public msrVoiceElement
```

More precisely and for technical reasons, an class `msrVoice` contains:

```
    // voice initial elements list

    std::list<S_msrVoiceElement>
                          fVoiceInitialElementsList;

    // voice first and last segments

    // fVoiceLastSegment contains the music
    // not yet stored in fVoiceInitialElementsList,
    // it is thus logically the end of the latter,
    // and is created implicitly for every voice.
    // It is needed 'outside' of the 'list<S_msrElement>'
    // because it is not a mere S_msrElement, but a S_msrSegment
    S_msrSegment          fVoiceLastSegment;

    // fVoiceFirstSegment is used to work around LilyPond_Issue_34
    S_msrSegment          fVoiceFirstSegment;
```

Each voice is described by a field of enumeration type `msrVoiceKind`, defined in src/formats/msr/msrBasicTypes.h:

```
enum class msrVoiceKind {
  kVoiceKindRegular,
  kVoiceKindDynamics,
  kVoiceKindHarmonies,  // for MusicXML <harmony/>, LilyPond ChordNames
  kVoiceKindFiguredBass // for MusicXML <figured-bass/>, LilyPond FiguredBass
};
```

As stated in the comment above, `fVoiceLastSegment` is used because it because `fVoiceInitialElementsList` can contain any class `msrVoiceElement`, whereas all MSR elements appended to the voice are to be placed in a segment.

An class `msrSegment` instance should thus be created and stored in `fVoiceLastSegment` before class `msrVoiceElement` instances can be appended to the voice.

When repeats are handled, an class msrRepeat instance is created. Then the contents of field msrVoice::fVoiceLastS
is moved into it and a new segment is created, see section 20.35 [Repeats], page 210.

Wether the last segment should be created right when the voice is created is controlled with enumeration type msrVoiceCreateInitialLastSegmentKind, defined in src/formats/msr/msrVoices.h:

```
enum class msrVoiceCreateInitialLastSegmentKind {
  kCreateInitialLastSegmentYes,
  kCreateInitialLastSegmentNo
};
```

20.29 Measures

Measures handling is presented at section 40 [Measures handling], page 288.

A measure is a linear, flat sequence of class msrMeasureElements, some of which are structured, such as class msrChord. Class msrMeausre is defined in src/formats/msr/msrMeausre.h/.cpp.

The measure elements are defined in src/formats/msr/:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msr > grep  'public
    msrMeasureElement' *.h
msrBars.h:class EXP msrBarCheck : public msrMeasureElement
msrBars.h:class EXP msrBarNumberCheck : public msrMeasureElement
msrBars.h:class EXP msrBarLine : public msrMeasureElement
msrBreaks.h:class EXP msrLineBreak : public msrMeasureElement
msrBreaks.h:class EXP msrPageBreak : public msrMeasureElement
msrClefs.h:class EXP msrClef : public msrMeasureElement
msrCodas.h:class EXP msrCoda : public msrMeasureElement
msrDoubleTremolos.h:class EXP msrDoubleTremolo : public msrMeasureElement
msrEyeGlasses.h:class EXP msrEyeGlasses : public msrMeasureElement
msrFiguredBasses.h:class EXP msrFiguredBass : public msrMeasureElement
msrHarmonies.h:class EXP msrHarmony : public msrMeasureElement
msrHiddenMeasureAndBarLines.h:class EXP msrHiddenMeasureAndBarLine : public
    msrMeasureElement
msrInstruments.h:class EXP msrScordatura : public msrMeasureElement
msrInstruments.h:class EXP msrAccordionRegistration : public msrMeasureElement
msrInstruments.h:class EXP msrHarpPedalsTuning : public msrMeasureElement
msrInstruments.h:class EXP msrPedal : public msrMeasureElement
msrInstruments.h:class EXP msrDamp : public msrMeasureElement
msrInstruments.h:class EXP msrDampAll : public msrMeasureElement
msrKeys.h:class EXP msrKey : public msrMeasureElement
msrLyrics.h:class EXP msrSyllable : public msrMeasureElement
msrMusicXMLSpecifics.h:class EXP msrPrintLayout : public msrMeasureElement
msrRehearsalMarks.h:class EXP msrRehearsalMark : public msrMeasureElement
msrSegnos.h:class EXP msrSegno : public msrMeasureElement
msrDalSegnos.h:class EXP msrDalSegno : public msrMeasureElement
msrStavesDetails.h:class EXP msrStaffDetails : public msrMeasureElement
msrTempos.h:class EXP msrTempo : public msrMeasureElement
msrTimeSignatures.h:class EXP msrTimeSignature : public msrMeasureElement
msrTranspositions.h:class EXP msrOctaveShift : public msrMeasureElement
msrTranspositions.h:class EXP msrTransposition : public msrMeasureElement
msrVoiceStaffChanges.h:class EXP msrVoiceStaffChange : public msrMeasureElement
```

In order to perform a time-wise analysis of the scores, MSR contains class msrmeasure linear flat lists, without the class msrRepeat and such being represented.
This is used when identifying rest notes that are not 'heard' simultaneously with other notes or rests: this way, the rest can ignore the current voice number and be placed in the vertical middle of the staff.

Apart from the cloning methods, only one method creates measures, namely
method msrSegment::createAMeasureAndAppendItToSegment (), defined in
src/formats/msr/msrSegments.h/.cpp:

```
S_msrMeasure msrSegment::createAMeasureAndAppendItToSegment (
  int         inputLineNumber,
  std::string measureNumber,
  msrMeasureImplicitKind
              measureImplicitKind)
{
  // ... ... ...

  ++gIndenter;

  // determine new measure 'first in segment' kind
  msrMeasureFirstInSegmentKind
    measureFirstInSegmentKind;

  if (fSegmentElementsList.size () == 0) {
    // this is the first measure in the segment
    measureFirstInSegmentKind =
      msrMeasureFirstInSegmentKind::kMeasureFirstInSegmentKindYes;
  }
  else {
    // this is not the first measure in the segment
    measureFirstInSegmentKind =
      msrMeasureFirstInSegmentKind::kMeasureFirstInSegmentKindNo;
  }

  // create a measure
  // ... ... ...

  S_msrMeasure
    result =
      msrMeasure::create (
        inputLineNumber,
        measureNumber,
        this);

  // set result's ordinal number
  result->
    setMeasureOrdinalNumberInVoice (
      fSegmentUpLinkToVoice->
        incrementVoiceCurrentMeasureOrdinalNumber ());

  // append result to the segment
  appendMeasureToSegment (result);

  --gIndenter;

  return result;
}
```

20.30 Repeats patterns and replicas

MSR represents repeated beats and measures this way:

- a pattern describes what is repeated;
- there are as many replicas of the music as needed.

This leads to:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msr > grep Pattern *.h | grep
    class
msrBeatRepeats.h:class EXP msrBeatRepeatPattern : public msrElement
msrMeasureRepeats.h:class EXP msrMeasureRepeatPattern : public msrElement
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msr > grep Replicas *.h | grep
    class
msrBeatRepeats.h:class EXP msrBeatRepeatReplicas : public msrElement
msrMeasureRepeats.h:class EXP msrMeasureRepeatReplicas : public msrElement
```

These two repeat cases are described in the sections below.

20.31 Beat repeats

Beat repeats handling is presented at section 45 [Beat repeats handling], page 321.

Class `msrBeatRepeat`, defined in `src/formats/msr/msrBeatRepeats.h/.cpp`, contains a pattern and replicas:

```
class EXP msrBeatRepeat : public msrVoiceElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrVoice              fUpLinkToBeatRepeatToVoice;

    // numbers
    int                     fBeatRepeatMeasuresNumber;
    int                     fBeatRepeatSlashesNumber;

    // measures repeat pattern
    S_msrBeatRepeatPattern
                            fBeatRepeatPattern;

    // measures repeat replicas
    S_msrBeatRepeatReplicas
                            fBeatRepeatReplicas;

    // measures repeat build phase, used when building the measures repeat
    msrBeatRepeatBuildPhaseKind
                            fCurrentBeatRepeatBuildPhaseKind; // unused??? JMI
};
```

Class `msrBeatRepeatPattern` contains a segment and an uplink:

```
class EXP msrBeatRepeatPattern : public msrElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrBeatRepeat       fUpLinkToBeatRepeat;

    // segment
    S_msrSegment          fBeatRepeatPatternSegment;
};
```

Class `msrBeatRepeatReplicas` contains a segment and an uplink:

```
class EXP msrBeatRepeatReplicas : public msrElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrBeatRepeat      fUpLinkToBeatRepeat;

    // segment
    S_msrSegment          fBeatRepeatReplicasSegment;
};
```

20.32 Measure repeats

Measure repeats handling is presented at section 46 [Measure repeats handling], page 322.

Class `msrMeasureRepeat`, defined in `src/formats/msr/msrMeasureRepeat.h/.cpp`, contains a pattern and replicas:

```
class EXP msrMeasureRepeat : public msrVoiceElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrVoice            fUpLinkToMeasureRepeatToVoice;

    // numbers
    int                   fMeasureRepeatMeasuresNumber;
    int                   fMeasureRepeatSlashesNumber;

    // measures repeat pattern
    S_msrMeasureRepeatPattern
                          fMeasureRepeatPattern;

    // measures repeat replicas
    S_msrMeasureRepeatReplicas
                          fMeasureRepeatReplicas;

    // measures repeat build phase, used when building the measures repeat
    msrMeasureRepeatBuildPhaseKind
                          fCurrentMeasureRepeatBuildPhaseKind;
};
```

Class `msrMeasureRepeatPattern` contains a segment and an uplink:

```
class EXP msrMeasureRepeatPattern : public msrElement
{
  // ... ... ...

 private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrMeasureRepeat   fUpLinkToMeasureRepeat;

    // segment
    S_msrSegment          fMeasureRepeatPatternSegment;
};
```

Class `msrMeasureRepeatReplicas` contain a segment and an uplink:

```
class EXP msrMeasureRepeatReplicas : public msrElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrMeasureRepeat   fUpLinkToMeasureRepeat;

    // segment
    S_msrSegment          fMeasureRepeatReplicasSegment;
};
```

20.33 Multi-measure rests

Multi-measure rests handling is presented at section 47 [Multi-measure rests handling], page 323.

Class `msrMultiMeasureRest`, defined in `src/formats/msr/msrMultiMeasureRest.h/.cpp`, essentially contains a liste of class `_msrMeasure` instances and a multi-measure rests number:

```
class EXP msrMultiMeasureRest : public msrSegmentElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    S_msrSegment          fMultiMeasureRestUpLinkToSegment;

    int                   fMeasureRestsNumber; // supplied by MusicXML
    std::list<S_msrMeasure>
                          fMeasureRestsList;

    int                   fLastMeasureRestPuristNumber;

    std::string           fNextMeasureNumber;
};
```

20.34 Barlines

20.35 Repeats

Repeats handling is presented at section 48 [Repeats handling], page 324.

Contrary to MusicXML, MusicFormats represents the full structure of repeated music, not just barlines.

The following classes are defined in src/formats/msr/msrRepeats.h/.cpp, contains:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msr > grep class   msrRepeats.h
class   msrRepeat;
class   msrMultiMeasureRest;
class   msrMeasureRepeat;
class   msrNote;
class EXP msrRepeatCommonPart : public msrElement
class EXP msrRepeatEnding : public msrElement
class EXP msrRepeat : public msrVoiceElement
class EXP msrRepeatDescr : public smartable
class EXP msrRepeatElement : public msrElement
```

Class msrRepeat, defined in msrBothmsrRepeats, contains an class msrRepeatCommonPart, followed by zero or more instances of class msrRepeatEnding:

```
class EXP msrRepeat : public msrVoiceElement
{
  public:

    // data types
    // ------------------------------------------------------

    enum class msrRepeatExplicitStartKind {
      kRepeatExplicitStartNo,
      kRepeatExplicitStartYes
    };

    // ... ... ...

    // common part
    S_msrRepeatCommonPart fRepeatCommonPart;

    // repeat endings
    std::vector<S_msrRepeatEnding>
                          fRepeatEndings;
    int                   fRepeatEndingsInternalCounter;

    // immediately preceding and following repeats
    // detecting several repeats in a row helps LilyPond code generation
    // depending on the options JMI
    S_msrRepeat           fImmediatelyPrecedingRepeat;
    S_msrRepeat           fImmediatelyFollowingRepeat;
};
```

Class msrRepeatCommonPart contains a list of class msrVoiceElement:

```
  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrRepeat           fRepeatCommonPartUpLinkToRepeat;

    // elements list
    std::list<S_msrVoiceElement>
                              fRepeatCommonPartElementsList;
```

Enumeration type `msrRepeatEndingKind` is used to distinguish hooked and hookless repeat endings: hookless when the ending is simply overlined, and hooked when there a vertical line at the end of the ending's overline:

```
enum class msrRepeatEndingKind {
  kRepeatEndingHooked,
  kRepeatEndingHookless
};
```

Class `msrRepeatEnding` contains a list of class `msrVoiceElement` too, as well as a enumeration type `msrRepeatEndingKind` field:

```
  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrRepeat               fRepeatEndingUpLinkToRepeat;

    // numbers
    std::string               fRepeatEndingNumber; // may be "1, 2"
    int                       fRepeatEndingInternalNumber; // internally assigned

    // kind
    msrRepeatEndingKind       fRepeatEndingKind;

    // elements list
    std::list<S_msrVoiceElement>
                              fRepeatEndingElementsList;
```

20.36 Segments

Segments handling is presented at section 44 [Segments handling], page 315.

Segment are not explicit in music scores, but they are there alright and we have to represent them in MSR:

- it is a sequence of music elements not containing a repeat. This is equivalent to so-called *basic blocs* in compiler technology, that are linear sequences of instructions without jumps, i.e. there is exactly one entry and one exit.

For example, at figure 20.4 [Three segments in a voice], page 212, there are three segments:

- the first one contains the `c1`, and belongs to a first repeat;
- the second one contains the `d1`, and is a member of the voice;
- the last one contains the `e1` and belongs to a second repeat.

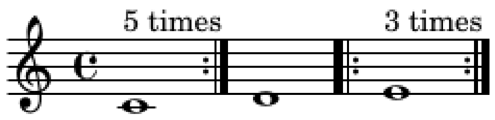


Figure 20.4: Three segments in a voice

20.37 Notes and rests

Class `msrNote` is complex class: it handles many variants, but using classes to represent the variants would be too cumbersone. As shown at figure 20.3 [The MSR classes hierarchy], page 199:

- a note can be a standalone (regular) note or rest;
- it can belong to a grace notes group;
- it can belong to chord, which can itself belong to a grace notes group or a tuplet;
- it can belong to a tuplet;
- it can belong to double tremolo;
- and finally, a rest can be unpiched.

class `msrNote` thus uses enumeration type enumeration type `msrNoteKind`, defined in `src/formats/msr/msrBasicTyp` to distinguish them:

```
enum class msrNoteKind {
  kNote_UNKNOWN_,

  // in measures
  kNoteRegularInMeasure,
  kNoteRestInMeasure,
  kNoteSkipInMeasure, // an invisible rest
  kNoteUnpitchedInMeasure,

  // in chords
  kNoteRegularInChord,

  // in tuplets
  kNoteRegularInTuplet,
  kNoteRestInTuplet,
  kNoteUnpitchedInTuplet,

  // in grace notes groups
  kNoteRegularInGraceNotesGroup,
  kNoteSkipInGraceNotesGroup, // used to circumvent LilyPond_Issue_34

  // in chords in grace notes groups
  kNoteInChordInGraceNotesGroup,

  // in tuplets in grace notes groups
  kNoteInTupletInGraceNotesGroup,

  // in double-tremolos
  kNoteInDoubleTremolo
};
```

20.38 Grace notes groups

Grace notes groups handling is presented at section 60 [Grace notes groups handling], page 344.

20.39 Chords

A chord contains notes only, and can occur in measures, tuplets and grace notes groups, hence:

```
// chords
//------------------------------------------------------------------------------

enum class msrChordInKind {
  kChordIn_UNKNOWN_,

  kChordInMeasure,
  kChordInTuplet,
  kChordInGraceNotesGroup
};
```

20.40 Tuplets

Tuplets handling is presented at section 62 [Tuplets handling], page 346.

A tuplet can contain:

- notes and rests;
- chords;
- other tuplets.

Tuplets can occur in measures and other tuplets, hence enumeration type `msrTupletInKind`:

```
enum class msrTupletInKind {
  kTupletIn_UNKNOWN_,

  kTupletInMeasure,
  kTupletInTuplet
};
```

Tuplets factors are represented by class `msrTupletFactor`, defined in `src/formats/msr/msrBasicTypes.h/.cpp`.

```
class EXP msrTupletFactor
{
  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    Bool                  isEqualToOne () const
                              {
                                return
                                  fTupletActualNotes == fTupletNormalNotes;
                              }

    mfRational               asRational () const
                               {
                               return
                                 mfRational (
                                   fTupletActualNotes,
                                   fTupletNormalNotes);
                             }

  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    int                      fTupletActualNotes;
    int                      fTupletNormalNotes;
};
```

20.41 Harmonies and figured bass similarities

Harmonies and figured bass handling is presented at section 63 [Harmonies handling], page 347 and section 64 [Figured bass elements handling], page 365, respectively.

In MusicXML, harmonies and figured bass occur at the measure level:

```
      <harmony print-frame="no">
        <root>
          <root-step>C</root-step>
          </root>
        <kind text="m">minor</kind>
        </harmony>
      <note default-x="75.17" default-y="-35.00">
        <pitch>
          <step>F</step>
          <octave>4</octave>
          </pitch>
        <duration>2</duration>
        <voice>1</voice>
        <type>quarter</type>
        <stem>up</stem>
        </note>
```

```
      <harmony>
        <root>
          <root-step>F</root-step>
          <root-alter>1</root-alter>
        </root>
        <kind>major</kind>
        <inversion>2</inversion>
      </harmony>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
```

In MSR, the instances of class `msrHarmony` and class `msrFiguredBass` are present twice:

- each class `msrNote` instance contains the harmonies and figured bass attached to it:

```
class EXP msrNote : public msrTupletElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // harmonies
    // ------------------------------------------------------

    std::list<S_msrHarmony>
                          fNoteHarmoniesList;

    // figured bass
    // ------------------------------------------------------

    std::list<S_msrFiguredBass>
                          fNoteFiguredBassesList;

  // ... ... ...
};
```

- each class `msrPart` instance contains a harmonies staff and voice, as well as a figured bass staff and voice:

```
class EXP msrPart : public msrPartGroupElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------
   // harmonies

    S_msrStaff            fPartHarmoniesStaff;
    S_msrVoice            fPartHarmoniesVoice;

    // figured bass

    S_msrStaff            fPartFiguredBassStaff;
    S_msrVoice            fPartFiguredBassVoice;

  // ... ... ...
};
```

The way harmonies and figured bass elements are represented in MusicFormats is presented in the next two sections.

20.42 Harmonies

Harmonies handling is presented at section 63 [Harmonies handling], page 347.

20.43 Figured bass

Figured bass elements handling is presented at section 64 [Figured bass elements handling], page 365.

20.44 Lyrics

Lyrics handling is presented at section 65 [Lyrics handling], page 381.

Lyrics are handled in rather a special way in music scores:

- they have a linear structure, independent of the repeats structure of the staff they belong too;
- the can be several lyrics stanzas associated to a given staff;
- the syllables in lyrics can apply to more that one note, and the subdivisions of words have to be handled.

The basic building block for lyrics in MSR is class `msrSyllable`, whose variants are distinguished by enumeration type enumeration type `msrSyllableKind`:

```
  enum class msrSyllableKind {
    kSyllableNone,
    kSyllableSingle,
    kSyllableBegin, kSyllableMiddle, kSyllableEnd,

    kSyllableOnRestNote,
    kSyllableSkipOnRestNote,
    kSyllableSkipOnNonRestNote,

    kSyllableMeasureEnd,
    kSyllableLineBreak, kSyllablePageBreak
  };
```

Extensions are described by enumeration type :

```
  enum class msrSyllableExtendKind {
    kSyllableExtendNone,
    kSyllableExtendEmpty,
    kSyllableExtendSingle,
    kSyllableExtendStart, kSyllableExtendContinue, kSyllableExtendStop
  };
```

Class `msrSyllable` contains:

```
  // syllable kind
  msrSyllableKind       fSyllableKind;

  // texts list
  std::list<std::string>
                        fSyllableElementsList;

  // extend kind
  msrSyllableExtendKind fSyllableExtendKind;

  // stanza number, may contain non-digits
  std::string           fSyllableStanzaNumber;

  // syllable whole notes
  mfRational              fSyllableWholeNotes;

  // syllable tuplet factor
  msrTupletFactor       fSyllableTupletFactor;
```

Syllables are one case where the data in MSR is denormalized: a given class `msrSyllable` instance belongs both to an class `msrNote` instance and to a lyrics instance of class `msrVoice`.

At the higher level, syllables are organized as instances of class `msrStanza`, which contains:

```
    // contents
    std::vector<S_msrSyllable> fSyllables;

    Bool                    fStanzaTextPresent;
```

20.45 MIDI

MIDI handling is presented at section 66 [MIDI handling], page 382.

Chapter 21

MSR time-oriented represention

In order to represent the music according to simultaneous sounding time, MSR builds:

- a flat list of measures at the voice and staff levels;
- from this, a vector of measures slices at the voice, staff, part, part group and score levels.

The source files are in `src/formats/msr/msrMeasuresSlices.h/.cpp`.

21.1 Note events

Notes start and stop are represented by enumeration type `msrNoteEventKind`:

```
enum class msrNoteEventKind {
  kNoteEventStart,
  kNoteEventStop
};
```

A note event is described in class :

```
class   msrNoteEvent : public smartable
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    mfRational              fNoteEventMeasurePosition;
    S_msrNote              fNoteEventNote;
    msrNoteEventKind       fNoteEventKind;
};
```

21.2 Simultaneous notes chunks

Such a chunk is a set of notes or rests played simultaneously, i.e. that start and stop at the same time. The set is stored as a list actually:

```
class   msrSimultaneousNotesChunk : public smartable
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    mfRational              fChunkMeasurePosition;
    std::list<S_msrNote>  fChunkNotesList;
    mfRational              fChunkNotesDurationWholeNotes;
};
```

21.3 Measures slices

A measures slice, described by class `msrMeasuresSlice`, is a 'vertical' cut in the score across voices: is contains all the measures starting at the same time, one per voice:

```
class EXP msrMeasuresSlice : public smartable
{
  // ... ... ...

  protected:

    // protected fields
    // ------------------------------------------------------

    int                   fSlicePuristMeasureNumber;
    std::string           fSliceMeasureNumber;

    // the measures in the slice
    std::vector<S_msrMeasure>  fSliceMeasuresVector;

    // notes flat list
    std::list<S_msrNote>  fSliceNotesFlatList;

    // note events list
    std::list<S_msrNoteEvent>
                          fSliceNoteEventsList;

    // simultaneous notes chunks list
    std::list<S_msrSimultaneousNotesChunk>
                          fSliceSimultaneousNotesChunksList;
};
```

21.4 Measures slices sequences

A class `msrMeasuresSlicesSequence` contains a vector of `S_msrMeasuresSlice` instances:

```
class EXP msrMeasuresSlicesSequence : public smartable
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    std::string           fMeasuresOrigin;

    std::vector<S_msrMeasuresSlice>
                          fMeasuresSlicesVector;
};
```

A smart pointer to am `msrMeasuresSlicesSequence` instance is stored in `msrVoice`, `msrStaff`, `msrPart`, `msrPartGroup` and `msrScore`.

21.5 Building the measures slices

21.5.1 Part measures slices

At the part level, this is done in method `msrPart::collectPartMeasuresSlices ()`:

```
void msrPart::collectPartMeasuresSlices (
  int inputLineNumber)
{
  // ... ... ...

  // create the part measures slices sequence
  fPartMeasuresSlicesSequence =
    msrMeasuresSlicesSequence::create (
      fPartName); // origin

  // populate it
  for (S_msrStaff staff : fPartAllStavesList) {
    // ... ... ...

    ++gIndenter;

    S_msrMeasuresSlicesSequence
      staffMeasuresSlicesSequence =
        staff->
          getStaffMeasuresSlicesSequence ();

    if (! staffMeasuresSlicesSequence) {
      std::stringstream ss;

      ss <<
        "The staffMeasuresSlicesSequence of staff \"" <<
        staff->getStaffName () <<
        "\" is null";

      musicxmlWarning (
        gServiceRunData->getInputSourceName (),
        inputLineNumber,
        ss.str ());
    }
    else {
      fPartMeasuresSlicesSequence->
        mergeWithMeasuresSlicesSequence (
          inputLineNumber,
          getPartCombinedName (),
          staffMeasuresSlicesSequence);
    }

    --gIndenter;
  } // for

  // ... ... ...
}
```

21.5.2 Staff measures slices

Method `msrStaff::collectStaffMeasuresSlices ()` builds them:

```
void msrStaff::collectStaffMeasuresSlices (
  int inputLineNumber)
{
  // ... ... ...

  // create the staff measures slices sequence
  fStaffMeasuresSlicesSequence =
    msrMeasuresSlicesSequence::create (
      fStaffName); // origin

  // populate it
  for (const S_msrVoice& voice : fStaffAllVoicesList) {
    // ... ... ...

    // get the voice measures slices sequence
    S_msrMeasuresSlicesSequence
      voiceMeasuresSlicesSequence =
        voice->
          getVoiceMeasuresSlicesSequence ();

    // merge it with the voice measures slices sequence
    if (voiceMeasuresSlicesSequence) { // JMI
      fStaffMeasuresSlicesSequence =
        fStaffMeasuresSlicesSequence->
          mergeWithMeasuresSlicesSequence (
            inputLineNumber,
            fStaffName,
            voiceMeasuresSlicesSequence);
    }

    // identify the solo notes and rests in the staff
    fStaffMeasuresSlicesSequence->
      identifySoloNotesAndRests ();

    --gIndenter;
  } // for

  // ... ... ...
}
```

21.6 Solo notes and rests

A solo note or rest is one that occurs alone at some point in time for its whole duration, without any other note being played at the same time.

Identifying such solo notes or rests is done in method `msrMeasuresSlicesSequence::identifySoloNotesAndRests (` using the measures slices of the staff they occur in, called method `msrStaff::collectStaffMeasuresSlices ()` as shown above:

```cpp
void msrMeasuresSlicesSequence::identifySoloNotesAndRests ()
{
  // ... ... ...

  // collect the notes from the sequence's measures slices
  for (
    std::vector<S_msrMeasuresSlice>::const_iterator i =
      fMeasuresSlicesVector.begin ();
    i != fMeasuresSlicesVector.end ();
    ++i
  ) {
    S_msrMeasuresSlice measuresSlice = (*i);

    measuresSlice->
      collectNonSkipNotesFromMeasuresSliceMeasures ();
  } // for
}
```

21.7 A measures slices example

Chapter 22

Path to voice

src/formats/msr/msrPathToVoice.h.h/.cpp defines class msrPathToVoice, used to create partial clones of class msrBook retaining only certain staves and/or voices, or to create new class msrScore instances containing each of them only:

```
class EXP msrPathToVoice : public smartable
{
  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    void                  appendPartGroup (const S_msrPartGroup& partGroup)
                              {
                                fPartGroupsList.push_back (partGroup);
                              }

  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    S_msrBook             fBook;

    S_msrScore            fScore;

    // part groups can be nested
    std::list<S_msrPartGroup>  fPartGroupsList;

    S_msrPart             fPart;

    S_msrStaff            fStaff;

    S_msrVoice            fVoice;
};
```

Chapter 23

LilyPond Scores Representation (LPSR)

An LPSR description contains two components:

- the first one is an MSR, thus the whole music score description;
- the second one is a description of the structure of the score mirroring LilyPond's specific blocks such as `\book {...}` and `\layout {...}` .

Class `lpsrScore` thus contains:

```
    // MSR data
    S_msrScore              fMsrScore;

    // ... ... ...

    // LilyPond stuff
    S_lpsrHeader            fScoreHeader;
    S_lpsrPaper             fScorePaper;
    S_lpsrLayout            fScoreLayout;

    // variables, voices and stanzas
    std::list<S_msrElement>
                            fScoreElementsList;

    // score LPSR book blocks list
    std::list<S_lpsrBookBlock> fScoreBookBlocksList;
    S_lpsrScoreBlock        fScoreScoreBlock; // JMI ???
```

23.1 LPSR basic types

Some types used thoughout LSPR are defined in src/formats/lpsr//lpsrEnumTypes.h/.cpp:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/lpsr > egrep -rIn '^// '
    lpsrEnumTypes.h
lpsrEnumTypes.h:28:// score output kinds
lpsrEnumTypes.h:50:// accidental styles
lpsrEnumTypes.h:53:// JMI there are new ones in LilyPond 2.22
lpsrEnumTypes.h:87:// chords languages
lpsrEnumTypes.h:107:// whole notes
lpsrEnumTypes.h:118:// dotted durations
lpsrEnumTypes.h:129:// rests measures
lpsrEnumTypes.h:135:// texts lists
lpsrEnumTypes.h:141:// pitches and octaves
lpsrEnumTypes.h:151:// lyrics durations
lpsrEnumTypes.h:168:// initialization
```

23.2 Adapting LilyPond code generation to the target version number

As of version 2.22, `compressMultiMeasureRests` has been replaced by `compressMeasureRests` for clarity.

Such is done specific methods:

Chapter 24

Braille Scores Representation (BSR)

BSR represents braille scores as composed of lines of 6-dot cells.

24.1 BSR basic types

Some types used thoughout BSR are defined in src/formats/bsr/bsrEnumTypes.h/.cpp:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/bsr >  egrep -rIn  '^// '
    bsrEnumTypes.h
bsrEnumTypes.h:23:// cell kinds
bsrEnumTypes.h:107:// lower case letters
bsrEnumTypes.h:139:// capitals
bsrEnumTypes.h:143://  kCellCapitalsSequenceSign, // { kCellCapitalsSign,
    kCellCapitalsSign };
bsrEnumTypes.h:145:// decimal digits
bsrEnumTypes.h:160:// lower decimal digits
bsrEnumTypes.h:174:// alterations
bsrEnumTypes.h:181:// augmentation dots
bsrEnumTypes.h:186:// arithmetic operators
bsrEnumTypes.h:195:// words
bsrEnumTypes.h:205:// braille cells
bsrEnumTypes.h:212:// braille output kinds
bsrEnumTypes.h:231:// chords languages
bsrEnumTypes.h:251:// brailling numbers
bsrEnumTypes.h:255:// brailling characters and strings
bsrEnumTypes.h:261:// writing UTF-16 to ostreams
bsrEnumTypes.h:273:// initialization
bsrEnumTypes.h:971:// constants
bsrEnumTypes.h:975:// computations
```

24.2 Representing cells

This is done basically with enumeration type enumeration type bsrCellKind, defined in src/formats/bsr/bsrEnumTy

```
// cell kinds
//______________________________________________________________________________
enum class bsrCellKind {
  kCellUnknown,

  // non 6dots values
  kCellEOL      , // L'\u000a'
  kCellEOP      , // L'\u000c'

  // 6dots values for Braille
  kDotsNone     , // L'\u2800'
  kDots1        , // L'\u2801'
  kDots2        , // L'\u2802'

  // ... ... ...

  kDots23456    , // L'\u283e'
  kDots123456     // L'\u283f'
};
```

Enumeration type `bsrCellKind` constants are not used throughout the code base: instead, there are enumeration type `bsrCellKind` constants to provide context-specific names for the cells kinds.

Lower-case letters:

```
// lower case letters
//______________________________________________________________________
const bsrCellKind
  kCellA = bsrCellKind::kDots1,
  kCellB = bsrCellKind::kDots12,


  kCellY = bsrCellKind::kDots13456,
  kCellZ = bsrCellKind::kDots1356;
```

Capital sign:

```
// capitals
//______________________________________________________________________
const bsrCellKind
  kCellCapitalsSign = bsrCellKind::kDots46;
```

Decimal digits:

```
// decimal digits
//______________________________________________________________________
const bsrCellKind
  kCellNumberSign = bsrCellKind::kDots3456,
  kCell1 = kCellA,
  kCell2 = kCellB,
  kCell3 = kCellC,
  kCell4 = kCellD,
  kCell5 = kCellE,
  kCell6 = kCellF,
  kCell7 = kCellG,
  kCell8 = kCellH,
  kCell9 = kCellI,
  kCell0 = kCellJ;
```

Alterations:

```
// alterations
//______________________________________________________________________
const bsrCellKind
  kCellFlat    = bsrCellKind::kDots126,
  kCellNatural = bsrCellKind::kDots16,
  kCellSharp   = bsrCellKind::kDots146;
```

Augmentation dots:

```
// augmentation dots
//______________________________________________________________________
const bsrCellKind
  kCellAugmentationDot = bsrCellKind::kDots3;
```

Arithmetic operators:

```
// arithmetic operators
//_______________________________________________________________________________
const bsrCellKind
  kCell_ac_plus      = bsrCellKind::kDots235,
  kCell_ac_minus     = bsrCellKind::kDots36,
  kCell_ac_times     = bsrCellKind::kDots35,
  kCell_ac_dividedBy = bsrCellKind::kDots25,
  kCell_ac_equals    = bsrCellKind::kDots2356;
```

Words:

```
// words
//_______________________________________________________________________________
const bsrCellKind
  kCellWordSign        = bsrCellKind::kDots345,

  kCellWordApostrophe  = bsrCellKind::kDots6,

  kCellParenthesis     = bsrCellKind::kDots2356,
  kCellQuestionMark    = bsrCellKind::kDots26;
```

Chapter 25

MusicXML Scores Representation (MXSR)

This format is provided by `libmusicxml2`, even though Dominique Fober didn't give it that name. It is a tree of class `mxmlelement` nodes, mapped one to one to the MusicXML markups.

The files in `libmusicxml/src`.

A set of interface functions is contained in `src/formats/mxsr/mxsr.h/.cpp`:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll formats/mxsr/
total 48
 0 drwxr-xr-x   6 jacquesmenu  staff   192 May 26 08:20:55 2021 ./
 0 drwxr-xr-x  10 jacquesmenu  staff   320 Jun 25 05:39:49 2021 ../
 8 -rw-r--r--@  1 jacquesmenu  staff  3292 Jun  6 06:35:19 2021 mxsr.cpp
 8 -rw-r--r--@  1 jacquesmenu  staff  1555 Jun  6 06:35:19 2021 mxsrGeneration.h
16 -rw-r--r--@  1 jacquesmenu  staff  7781 Jun  6 06:35:19 2021 mxsr0ah.cpp
16 -rw-r--r--@  1 jacquesmenu  staff  4829 Jun  6 06:35:19 2021 mxsr0ah.h
```

25.1 MusicXML elements and attributes

MusicXML data contains so-called elements, written as `<... />` markups, that can be nested:

```
      <system-margins>
        <left-margin>15</left-margin>
        <right-margin>0</right-margin>
      </system-margins>
```

In the example above, the values of the two margins are `15` and `0`, respectively.

MusicXML elements can have attributes, such as `version` below:

```
<score-partwise version="3.1">
```

The values of the elements and attributes are strings.

There are two special elements at the beginning of MusicXML data:

- a `<?xml/>` element indicating the characters encoding used;
- a <"!"`DOCTYPE/>` element telling that the contents is in 'score-partwise' mode and containing the URL of the DTD.

An exemple is:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 2.0 Partwise//EN"
                                "http://www.musicxml.org/dtds/partwise.dtd">
```

25.2 The xmlelement and xmlattribute types

`xmlelementxmlattribute`

These two classes are defined in `libmusicxml/src/elements!xml.h/.cpp`:

```
class   xmlelement;
class   xmlattribute;

typedef SMARTP<xmlattribute>  Sxmlattribute;
typedef SMARTP<xmlelement>    Sxmlelement;
```

class `xmlattribute` contains:

```
//______________________________________________________________________________
class EXP xmlattribute : public smartable {
  //! the attribute name
  std::string fName;
  //! the attribute value
  std::string   fValue;

  // ... ... ...

    //! returns the attribute value as a int
    operator int () const;
    //! returns the attribute value as a long
    operator long () const;
    //! returns the attribute value as a float
    operator float () const;
```

class `xmlelement` contains:

```
class EXP xmlelement : public ctree<xmlelement>, public visitable
{
  private:
    //! the element name
    std::string fName;
    //! the element value
    std::string fValue;
    //! list of the element attributes
    std::vector<Sxmlattribute> fAttributes;

  protected:
    // the element type
    int fType;
    // the input line number for messages to the user
    int fInputStartLineNumber;

  // ... ... ...

    //! returns the element value as a long
    operator long () const;
    //! returns the element value as a int
    operator int () const;
    //! returns the element value as a float
    operator float () const;
    //! elements comparison
    Bool operator ==(const xmlelement& elt) const;
    Bool operator !=(const xmlelement& elt) const { return !(*this == elt); }

    //! adds an attribute to the element
    long add (const Sxmlattribute& attr);

  // ... ... ...
};
```

Type `Sxmlelement` is a smart pointer to an `xmlelement`, so it is an `xmlelement` tree, since `xmlelement` is a recursive type.

`fInputStartLineNumber` is used for example in warning and error messages, to help the user locate the problem.

`fType` typically contains a value of some enumeration type , more on this below.

25.3 Enumeration types for `xmlelement`'s `fType`

`xmlelement`

`libmusicxml2` uses `elements/templates/elements.bash`, a Bash script, to generate the enumeration type constants and classes source code from the MusicXML DTD. This is not done in the `Makefile`, since it is to be run by hand only once.

The DTD files we use as reference are in `libmusicxml/dtds/3.1/schema`;

```
jacquesmenu@macmini: ~/musicformats-git-dev/libmusicxml/dtds/3.1/schema > ls -sal *.mod
 40 -rwxr-xr-x@ 1 jacquesmenu  staff  20238 Apr 22 15:49 attributes.mod
 16 -rwxr-xr-x  1 jacquesmenu  staff   4943 Apr 22 15:49 barLine.mod
 80 -rwxr-xr-x@ 1 jacquesmenu  staff  37932 Apr 22 15:49 common.mod
 88 -rwxr-xr-x@ 1 jacquesmenu  staff  41960 Apr 22 15:49 direction.mod
 16 -rwxr-xr-x@ 1 jacquesmenu  staff   4097 Apr 22 15:49 identity.mod
 24 -rwxr-xr-x@ 1 jacquesmenu  staff  10266 Apr 22 15:49 layout.mod
  8 -rwxr-xr-x@ 1 jacquesmenu  staff   2833 Apr 22 15:49 link.mod
104 -rwxr-xr-x@ 1 jacquesmenu  staff  51384 Apr 22 15:49 note.mod
 32 -rwxr-xr-x@ 1 jacquesmenu  staff  15476 Apr 22 15:49 score.mod
```

The first result of running `libmusicxml/src/elements/templates!elements.bash` is an anonymous enumeration type defined in `libmusicxml/src/elements!elements.h`:

```
enum class {
  kNoElement,
  kComment,
  kProcessingInstruction,
  k_accent,
  k_accidental,
  k_accidental_mark,
  k_accidental_text,

  // ... ... ...

  k_work,
  k_work_number,
  k_work_title,
  kEndElement
};
```

The constants `kNoElement`, `kComment` and `kProcessingInstruction` are added by `elements.bash`.

25.4 Classes for the `xmlelements`

All the MusicXML classes are instantiated from the `musicxml` template class, defined in libmusicxml/src/elements!
This is where `fType` gets its value:

```
template <int elt> class   musicxml : public xmlelement
{
  protected:
    musicxml (int inputLineNumber) : xmlelement (inputLineNumber) { fType = elt; }
};
```

The smart pointer s to the various elements are defined in libmusicxml/src/elements!typedef.h, using an anonymous enumeration type :

```
typedef SMARTP<musicxml<kComment> >          S_comment;
typedef SMARTP<musicxml<kProcessingInstruction> > S_processing_instruction;

typedef SMARTP<musicxml<k_accent> >   S_accent;
typedef SMARTP<musicxml<k_accidental> >   S_accidental;
typedef SMARTP<musicxml<k_accidental_mark> >    S_accidental_mark;
typedef SMARTP<musicxml<k_accidental_text> >    S_accidental_text;

// ... ... ...

typedef SMARTP<musicxml<k_work> >   S_work;
typedef SMARTP<musicxml<k_work_number> >    S_work_number;
typedef SMARTP<musicxml<k_work_title> >   S_work_title;
```

The two-way correspondance of MusicXML elements names to type `Sxmlelement` is stored `fMap` and `fType2Name`, defined in libmusicxml/src/elements!factory.h:

```
class EXP factory : public singleton<factory>{

  std::map<std::string, functor<Sxmlelement>*> fMap;
  std::map<int, const char*>  fType2Name;

  // ... ... ...
};
```

Those two maps are initialized in libmusicxml/samples/elements/factory.cpp:

```
factory::factory()
{
  fMap["comment"]      = new newElementFunctor<kComment>;
  fMap["pi"]         = new newElementFunctor<kProcessingInstruction>;
  fType2Name[kComment]   = "comment";
  fType2Name[kProcessingInstruction]   = "pi";

  fMap["accent"]   = new newElementFunctor<k_accent>;
  fMap["accidental"]   = new newElementFunctor<k_accidental>;
  fMap["accidental-mark"]    = new newElementFunctor<k_accidental_mark>;
  fMap["accidental-text"]    = new newElementFunctor<k_accidental_text>;

  // ... ... ...

  fMap["work"]   = new newElementFunctor<k_work>;
  fMap["work-number"]    = new newElementFunctor<k_work_number>;
  fMap["work-title"]   = new newElementFunctor<k_work_title>;

  fType2Name[k_accent]   = "accent";
  fType2Name[k_accidental]   = "accidental";
  fType2Name[k_accidental_mark]    = "accidental-mark";
  fType2Name[k_accidental_text]    = "accidental-text";

  // ... ... ...

  fType2Name[k_work]  = "work";
  fType2Name[k_work_number]   = "work-number";
  fType2Name[k_work_title]  = "work-title";
}
```

class `newElementFunctor` is defined in to provide call operator as:

```
template<int elt>
class   newElementFunctor : public functor<Sxmlelement>
{
  public:

    Sxmlelement operator ()()
        { return musicxml<elt>::new_musicxml (libmxmllineno); }
};
```

25.5 xmlelement trees

This section describes features supplied by `libmusicxml2`.

An `xmlelement` is the basic brick to represent a MusicXML element.

Smart pointer type `SXMLFile` is defined in `libmusicxml/src/xmlfile.h`:

```
//______________________________________________________________________________
class EXP TXMLFile : public smartable
{
  private:
    TXMLDecl*             fXMLDecl;
    TDocType*             fDocType;
    Sxmlelement           fXMLTree;

  protected:
       TXMLFile () : fXMLDecl(0), fDocType(0) {}
    virtual ~TXMLFile () { delete fXMLDecl; delete fDocType; }

  public:
    static SMARTP<TXMLFile> create();

  public:
    TXMLDecl*     getXMLDecl ()     { return fXMLDecl; }
    TDocType*     getDocType ()     { return fDocType; }
    Sxmlelement   elements ()       { return fXMLTree; }

    void      set (Sxmlelement root)  { fXMLTree = root; }
    void      set (TXMLDecl * dec)  { fXMLDecl = dec; }
    void      set (TDocType * dt)   { fDocType = dt; }

    void      print (std::ostream& s);
};
typedef SMARTP<TXMLFile> SXMLFile;
```

25.5.1 Creating xmlelement trees from textual data

Reading MusicXML data creates instances of xmlelement. This is done by and instance of xmlreader, defined in libmusicxml/src/xmlreader.h/.cpp, which provides methods:

```
    SXMLFile readbuff(const char* file);
    SXMLFile read(const char* file);
    SXMLFile read(FILE* file);
```

These three functions are defined this way:

```
//______________________________________________________________________________
SXMLFile xmlreader::readbuff(const char* buffer)
{
  fFile = TXMLFile::create();
  debug("read buffer", '-');
  return readbuffer (buffer, this) ? fFile : 0;
}

//______________________________________________________________________________
SXMLFile xmlreader::read(const char* file)
{
  fFile = TXMLFile::create();
  debug("read", file);
  return readfile (file, this) ? fFile : 0;
}

//______________________________________________________________________________
SXMLFile xmlreader::read(FILE* file)
{
  fFile = TXMLFile::create();
  return readstream (file, this) ? fFile : 0;
}
```

25.5.2 Printing xmlelement trees

An xmlelement can be printed by function printMxsr (), defined in src/formats/mxsr/mxsr.h/.cpp:

```
void printMxsr (const Sxmlelement theMxsr, std::ostream& os)
{
  xmlvisitor v (os);
  tree_browser<xmlelement> browser (&v);
  browser.browse (*theMxsr);
}
```

This how MusicXML and Guido output are generated.

25.6 The SXMLFile type

SXMLFile is defined in libmusicxml/src/factory!musicxmlfactory.h as a smart pointer to class TXMLFile:

```
//______________________________________________________________________________
class EXP TXMLFile : public smartable
{
  private:
    TXMLDecl*           fXMLDecl;
    TDocType*           fDocType;
    Sxmlelement         fXMLTree;

  protected:
      TXMLFile () : fXMLDecl(0), fDocType(0) {}
    virtual ~TXMLFile () { delete fXMLDecl; delete fDocType; }

  public:
    static SMARTP<TXMLFile> create();

  public:
    TXMLDecl*     getXMLDecl ()     { return fXMLDecl; }
    TDocType*     getDocType ()     { return fDocType; }
    Sxmlelement   elements ()       { return fXMLTree; }

    void      set (Sxmlelement root)  { fXMLTree = root; }
    void      set (TXMLDecl * dec)  { fXMLDecl = dec; }
    void      set (TDocType * dt)   { fDocType = dt; }

    void      print (std::ostream& s);
};
typedef SMARTP<TXMLFile> SXMLFile;
```

fXMLDecl describes the <?xml/> element and fDocType contains the <"!"DOCTYPE/> element.

Part VIII

Passes

Chapter 26

The passes

A pass performs a single translation from one music score description into another, such as from MusicXML to an MXSR, or from an MXSR to an MSR. The name 'pass' comes from the compiler writing field.

26.1 Translating MusicXML data to an MXSR format

This is supplied by the `libmusicxml2` library, a version of which is distributed as part of MusicFormats to avoid the need of two installs and the potential associated problems.

26.1.1 MusicXML coverage

`src/passes/mxsr2msr/mxsr2msrSkeletonBuilder.h/.cpp` and `src/passes/msr2mxsr/msr2mxsrTranslator.h/.c` handle most of the MusicXML version 3.1 elements. Some of them are handled by both sub-passes, such as `S_voice`, `S_measure` and `S_harmony`.

Among the elements that MusicFormats does not handled are the ones for which there is no occurrence in the corpus in folder `files/musicxml`, such as `beat-unit-tied` and `metronome-tied`.

The elements that are new in MusicXML version 4.0 are not known nor handled yet as of this writing.

26.1.2 `Backup` and `forward` handling

These markups move the 'MusicXML `drawing` point' left and right. Class `mxsr2msrTranslator` does this this way:

26.2 Translating an MXSR to an MSR

This is done by class `mxsr2msrTranslator`.

26.3 Translating an MSR to an MXSR

26.4 Translating an MSR to another MSR

Such translation is meant to offer an opportunity to modify the score's description depending on options.

26.5 Translating an MSR to an LPSR

This converter embeds a specific converter of MSR to MSR, to circumvent the famous LilyPond issue #34.

26.6 Translating an LPSR to LilyPond code

There are two visiting trace options for the generation of LilyPond code, one for its MSR component, and the other one for its LPSR own part:

```
// %%%JMI    Bool                    fGenerateMsrVisitingInformation;
    Bool                    fGenerateLpsrVisitingInformation;
```

26.7 Translating an MSR to an BSR

This converter embeds a specific converter of MSR to MSR, to circumvent the famous LilyPond issue #34.

26.8 Translating a BSR to another BSR

26.9 Translating an MXSR to Guido

Chapter 27

LilyPond code generation

LilyPond code is produced on standard output, unless options option `-output-file-name, -obj` or option `-auto-output-file-name, -aofn` are used.

27.1 Basic principle

Lilypond generation is done in `src/passes/lpsr2lilypond/lpsr2lilypondTranslator.h/.cpp`.

Class `lpsr2lilypondTranslator` provides:

```
public:

                        lpsr2lilypondTranslator (
                          const S_msrOahGroup&   msrOpts,
                          const S_lpsrOahGroup&  lpsrOpts,
                          std::ostream&          lilypondCodeStream);

  virtual               ~lpsr2lilypondTranslator ();

  void                  translateLpsrToLilypondCode (
                          const S_lpsrScore& lpsrScore);
```

It contains these fields among others:

```
private:

  // options
  // ------------------------------------------------------

  S_msrOahGroup          fMsrOahGroup;
  S_lpsrOahGroup         fLpsrOahGroup;

  // the LPSR score we're visiting
  // ------------------------------------------------------
  S_lpsrScore            fVisitedLpsrScore;

  // the output stream
  // ------------------------------------------------------

  std::ostream&              fLilypondCodeStream;
```

27.2 Generating Scheme functions in the LilyPond output

`xml2ly` can generate Scheme code that is used by the LilyPond code it generates. This is described in class `lpsrScore` by a number of `*IsNeeded` fields, such as:

```
    // files includes
    Bool                    fJianpuFileIncludeIsNeeded;

    // Scheme modules
    Bool                    fScmAndAccregSchemeModulesAreNeeded;

    // Scheme functions
    Bool                    fTongueSchemeFunctionIsNeeded;
```

Chapter 28

Braille generation

Braille is written to standard output or to files as binary data. Our reference is http://www.brailleauthority.org/music/Music_Braille_Code_2015.pdf.

28.1 Basic principle

Lilypond generation is done in src/passes/bsr2braille/bsr2brailleTranslator.h/.cpp.

Class `bsr2brailleTranslator` provides:

```
public:

                        bsr2brailleTranslator (
                          const S_bsrScore&     bsrScore,
                          const S_bsrOahGroup& bsrOpts,
                          std::ostream&        brailleOutputStream);

  virtual               ~bsr2brailleTranslator ();

  void                  translateBsrToBraille ();
```

It contains these fields among others:

```
private:

  // options
  // ------------------------------------------------------

  S_bsrOahGroup         fBsrOahGroup;

  // the BSR score we're visiting
  // ------------------------------------------------------

  S_bsrScore            fVisitedBsrScore;

  // the braille generator used
  // ------------------------------------------------------

  S_bsrBrailleGenerator fBrailleGenerator;

  // the output stream
  // ------------------------------------------------------

  std::ostream&              fBrailleOutputStream;
```

28.2 Output files name and contents options

he contents options use the following enumeration types:

```
enum class bsrUTFKind {
  kUTF8, kUTF16
};

enum class bsrByteOrderingKind {
  kByteOrderingNone,
  kByteOrderingBigEndian, kByteOrderingSmallEndian
};
```

`xml2brl` supplies a option `-files` options subgroup:

```
jacquesmenu@macmini > xml2brl -query files
--- Help for subgroup "files" in group "Files group" ---
  Files group (-files-group):
  --------------------------
    Files (-files):
      -obj, -output-file-name FILENAME
            Write Braille to file FILENAME instead of standard output.
      -aofn, -auto-output-file-name
            This option can only be used when reading from a file.
            Write MusicXML code to a file in the current working directory.
            The file name is derived from that of the input file,
            replacing any suffix after the '.' by 'xml'
            or adding '.xml' if none is present.
      -bok, -braille-output-kind OUTPUT_KIND
            Use OUTPUT_KIND to write the generated Braille to the output.
            The 4 output kinds available are:
            ascii, utf16, utf8 and utf8d.
            'utf8d' leads to every line in the braille score to be generated
            as a line of cells followed by a line of text showing the contents
            for debug purposes.
            The default is 'ascii'.
      -ueifn, -use-encoding-in-file-name
            Append a description of the encoding used
            and the presence of a BOM if any to the file name before the '.'.
      -bom, -byte-ordering-mark BOM_ENDIAN
            Generate an initial BOM_ENDIAN byte ordering mark (BOM)
            ahead of the Braille nusic code,
            which can be one of 'big' or 'small'.
            By default, a big endian BOM is generated.
```

28.3 Braille generators

The following classes are defined in src/formatsgeneration/brailleGeneration/brailleGeneration.h/.cpp

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formatsgeneration/brailleGeneration > grep
     class   brailleGeneration.h
enum class bsrUTFKind {
enum class bsrByteOrderingKind {
class EXP bsrBrailleGenerator : public smartable
/* this class is purely virtual
class EXP bsrAsciiBrailleGenerator : public bsrBrailleGenerator
class EXP bsrUTF8BrailleGenerator : public bsrBrailleGenerator
class EXP bsrUTF8DebugBrailleGenerator : public bsrUTF8BrailleGenerator
class EXP bsrUTF16BigEndianBrailleGenerator : public bsrBrailleGenerator
class EXP bsrUTF16SmallEndianBrailleGenerator : public bsrBrailleGenerator
```

The base class `bsrBrailleGenerator` contains:

```
public:

  // public services
  // ------------------------------------------------------

  virtual void          generateCodeForBrailleCell (
                          bsrCellKind cellKind) = 0;

  void                  generateCodeForCellsList (
                          const S_bsrCellsList& cellsList);

  virtual void          generateCodeForMusicHeading (
                          const S_bsrMusicHeading& musicHeading);

  virtual void          generateCodeForLineContents (
                          const S_bsrLineContents& lineContents);

  // ... ... ...

protected:

  // protected fields
  // ------------------------------------------------------

  std::ostream&             fBrailleOutputStream;
```

28.4 Writing braille cells

Braille cells are output to an `std::ostream` as hexadecimal strings by virtual method `generateCodeForBrailleCell`
methods in src/passes/bsr2braille/brailleGeneration.h, depending on the kind of output selected.

For example, ASCII braille generation is done by:

```
void bsrAsciiBrailleGenerator::generateCodeForBrailleCell (
  bsrCellKind cellKind)
{
  std::string stringForCell;

  switch (cellKind) {
    case bsrCellKind::kCellUnknown:
      {
        std::stringstream ss;

        ss <<
          "cannot generate code for braille cell '" <<
          bsrCellKindAsString (cellKind) <<
          "'";
        msrInternalError (
          gServiceRunData->getInputSourceName (),
          -98, // inputLineNumber, TICINO JMI
          __FILE__, __LINE__,
          ss.str ());
      }
      break;

    case bsrCellKind::kCellEOL:    stringForCell = "\x0a"; break;
    case bsrCellKind::kCellEOP:    stringForCell = "\x0c"; break;

    case bsrCellKind::kDotsNone:   stringForCell = "\x20"; break;

    case bsrCellKind::kDots1:      stringForCell = "\x41"; break;
    case bsrCellKind::kDots2:      stringForCell = "\x31"; break;

    // ... ... ...

    case bsrCellKind::kDots23456:  stringForCell = "\x29"; break;
    case bsrCellKind::kDots123456: stringForCell = "\x3d"; break;
  } // switch

  fBrailleOutputStream <<
    stringForCell;
}
```

Chapter 29

MusicXML generation

MusicXML text is produced on the standard output stream, unless options '-output-file-name' or '-auto-output-f
are used.

29.1 Basic principle

MusicXML generation is done in two passes:

- first create and MXSR containing the data;
- then simply write this tree.

29.2 Creating an xmlelement

An simple example is:

```
  // create a direction element
  Sxmlelement directionElement = createMxmlelement (k_direction, "");

  // set it's "placement" attribute if relevant
  std::string
    placementString =
      msrPlacementKindAsMusicXMLString (placementKind);

  if (placementString.size ()) {
    directionElement->add (createMxmlAttribute ("placement", placementString));
  }
```

This one supplies a value to the xmlelement it creates:

```
void msr2mxsrTranslator::visitStart (S_msrIdentification& elt)
{
  // composers
  const std::list<std::string>&
    composersList =
      elt->getComposersList ();

  for (
    std::list<std::string>::const_iterator i = composersList.begin ();
    i != composersList.end ();
    ++i
  ) {
    std::string variableValue = (*i);

    // create a creator element
    Sxmlelement creatorElement = createMxmlelement (k_creator, variableValue);

    // set its "type" attribute
    creatorElement->add (createMxmlAttribute ("type", "composer"));

    // append it to the composers elements list
    fComposersElementsList.push_back (creatorElement);
  } // for

  // ... ... ...
}
```

29.3 Creating an xmlelement tree

In , this code:

```
void msr2mxsrTranslator::visitStart (S_msrClef& elt)
{
    // ... ... ...

    Sxmlelement clefElement = createMxmlelement (k_clef, "");

    // set clefElement's "number" attribute if relevant
    /*
      0 by default in MSR,
      1 by default in MusicXML:
        The optional  number attribute refers to staff numbers within the part,
        from top to bottom on the system.
        A value of 1 is assumed if not present.
    */

    int clefStaffNumber =
      elt->getClefStaffNumber ();

    if (clefStaffNumber > 1) {
      clefElement->add (
        createMxmlIntegerAttribute ("number", clefStaffNumber));
    }

    // populate clefElement
    switch (elt->getClefKind ()) {
      // ... ... ...

      case msrClefKind::kClefTrebleMinus8:
        {
          clefElement->push (
            createMxmlelement (
              k_sign,
              "G"));
          clefElement->push (
            createMxmlIntegerElement (
              k_line,
              2));
          clefElement->push (
            createMxmlIntegerElement (
              k_clef_octave_change,
              -1));
        }
        break;

      // ... ... ...
}
```

creates this MusicXML element depending on the value returned by method `msrClef::getClefStaffNumber ()`:

```
<clef number="2">
     <sign>G</sign>
     <line>2</line>
     <clef-octave-change>-1</clef-octave-change>
 </clef>
```

29.4 Browsing the visited MSR score

The creation of the tree is done in src/passes/msr2mxsr/msr2mxsrTranslator.h/.cpp.

Class `msr2mxsrTranslator` is defined in those files, it contains:

```
public:

                        msr2mxsrTranslator (
                          const S_msrScore& visitedMsrScore);

  virtual               ~msr2mxsrTranslator ();

  Sxmlelement           translateMsrToMxsr ();

// ... ... ...

private:

  // the MSR score we're visiting
  // ------------------------------------------------------
  S_msrScore                fVisitedMsrScore;


  // the MXSR we're building
  // ------------------------------------------------------
  Sxmlelement               fResultingMusicxmlelement;
```

The method `msr2mxsrTranslator::translateMsrToMxsr ()` method does the following:

```
//________________________________________________________________________
Sxmlelement msr2mxsrTranslator::translateMsrToMxsr ()
{
#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  mfAssert (
    __FILE__, __LINE__,
    fVisitedMsrScore != nullptr,
    "fVisitedMsrScore is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  // create the current score part-wise element
  fResultingMusicxmlelement =
    createMxmlScorePartWiseElement ();

  // create a msrScore browser
  msrBrowser<msrScore> browser (this);

  // set the parts browsing order
  theMsrScore->
    setStavesBrowingOrderKind (
      msrStavesBrowingOrderKind::kStavesBrowingOrderHarmoniesFiguredBassesRegulars);

  // browse the visited MSR score
  browser.browse (*fVisitedMsrScore);

  return fResultingMusicxmlelement;
}
```

29.5 Ancillary functions to create MXSR data

The function `createMxmlScorePartWiseElement ()` is defined in `src/formats/mxsr/mxsr.h/.cpp`:

```
//------------------------------------------------------------------------
Sxmlelement createMxmlScorePartWiseElement ()
{
  Sxmlelement result = factory::instance ().create (k_score_partwise);

  Sxmlattribute versionAttribute = createMxmlAttribute("version", "3.1");
  result->add (versionAttribute);

  return result;
}
```

Chapter 30

Guido code generation

Guido code is produced on standard output, unless options option `-output-file-name, -obj` or option `-auto-output-file-name, -aofn` are used.

30.1 Basic principle

As is done for MusicXML generation, Guido generation is done in two passes:

- first create and `mxsr` containing the data;
- then simply write this tree.

The creation of the tree is done in `src/passes/msr2mxsr/msr2mxsrTranslator.h/.cpp`. See subsection 29.1 [musicxmlGeneration], page 245, for more details.

Part IX

Generators

Chapter 31

The generators

A generator creates a music score ex-nihilo, without any description of the music being input. It's behaviour can be adapted to the users needs with options if needed.

Generators are supplied in the `src/generators/` directory. They don't have any interface in at the time of this writing, even though they could.

31.1 MusicAndHarmonies

`MusicAndHarmonies.cpp`

31.2 Mikrokosmos3Wandering

This service produces the score for Zoltán Kodály's Mikrokosmos III Wandering score, taking inspiration from the same example in Abjad (`http://abjad.mbrsi.org/literature_examples/bartok.html`). Is was written in the first place to check the MSR API before writing the MSDL converter.

The score produced is shown at figure 31.1 [Zoltán Kodály's Mikrokosmos III Wandering], page 251.

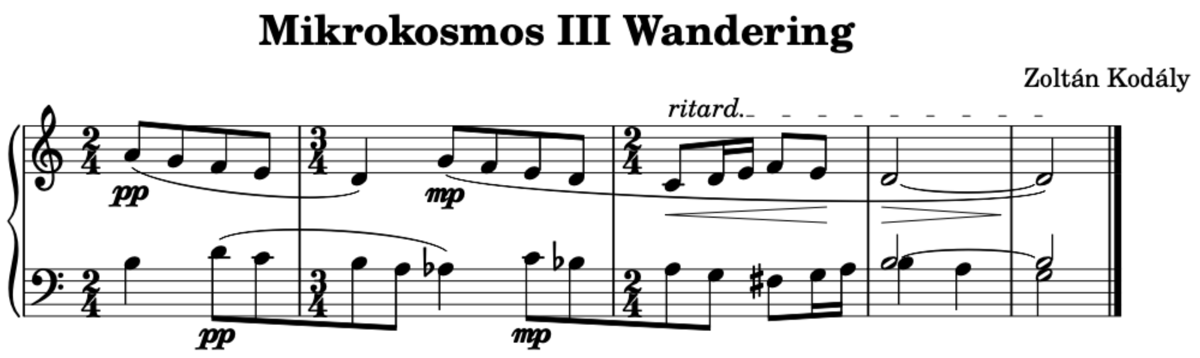


Figure 31.1: Zoltán Kodály's Mikrokosmos III Wandering

31.3 LilyPondIssue34

This service produces the same score as that obtained by:

```
xml2ly -auto-output-file-name gracenotes/LilyPondIssue34.xml
```

The resulting score is shown at figure 31.2 [The LilyPondIssue34 score], page 252.

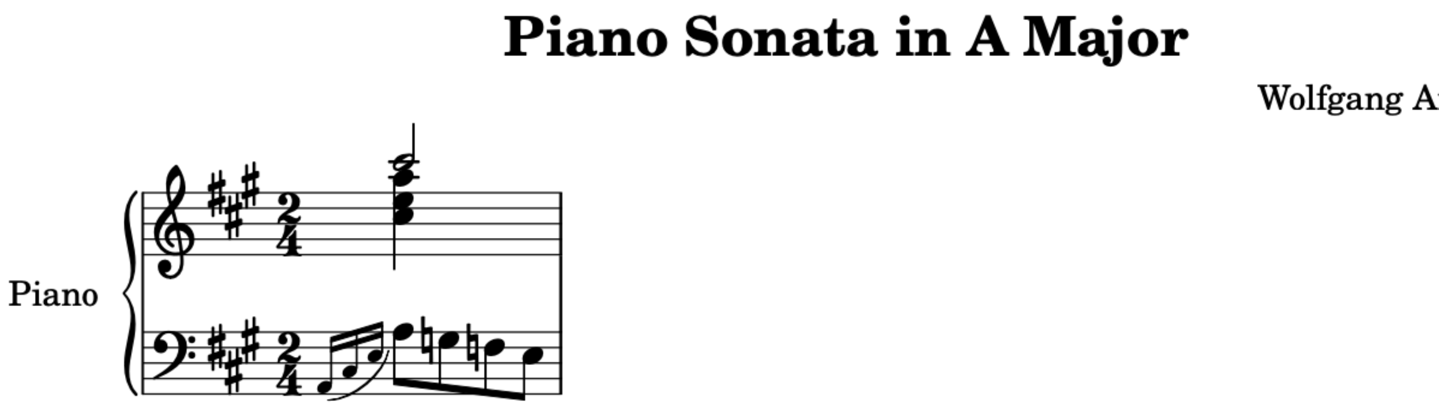


Figure 31.2: The LilyPondIssue34 score

The name `LilyPondIssue34` stems from the fact that translating this MusicXML file to LilyPond with `musicxml2ly` exhibits the famous LilyPond issue #34.

This example was written to design a LilyPond-oriented interface to LPSR, preparing the grounds for LilyPond export to other formats. This work in in progress at the time of this writing.

Part X

Converters

Chapter 32

The converters

A multi-pass converter performs a sequence of passes, i.e. a sequence of steps. For example, `xml2ly` performs the following passes:

```
jacquesmenu@macmini: ~/musicformats-git-dev/musicxml > xml2ly -about
What xml2ly does:

    This multi-pass converter basically performs 5 passes:
        Pass 1:  reads the contents of MusicXMLFile or stdin ('-')
                 and converts it to a MusicXML tree;
        Pass 2a: converts that MusicXML tree into
                 a first Music Score Representation (MSR) skeleton;
        Pass 2b: populates the first MSR skeleton from the MusicXML tree
                 to get a full MSR;
        Pass 3:  converts the first MSR into a second MSR to apply options
        Pass 4:  converts the second MSR into a
                 LilyPond Score Representation (LPSR);
        Pass 5:  converts the LPSR to LilyPond code
                 and writes it to standard output.

    Other passes are performed according to the options, such as
    displaying views of the internal data or printing a summary of the score.

    The activity log and warning/error messages go to standard error.
```

32.1 `xml2ly`

MusicXML (*Music eXtended Markup Language*) is a specification language meant to represent music scores by texts, readable both by humans and computers. It has been designed by the W3C Music Notation Community Group (`https://www.w3.org/community/music-notation/`) to help sharing music score files between applications, through export and import mechanisms.

The homepage to MusicXML is `https://www.musicxml.com`.

MusicXML data contains very detailed information about the music score, and it is quite verbose by nature. This makes creating such data by hand quite difficult, and this is done by applications actually.

The MusicXML data is not systematically checked for correctness. Checks are done, however, to ensure it won't crash due to missing values.

32.2 `xml2brl`

`xml2brl` is mentioned here, but not described in detail.

32.3 xml2xml

32.4 xml2gmn

32.5 msdlconverter

Part XI

Interfaces

Chapter 33

Library interfaces

Chapter 34

Representations interfaces

These interfaces are a set of functions to create formats for various needs.

34.1 MSR interfaces

The MSR interfaces are in `interfaces/msrinterfaces/`:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll interfaces/msrinterfaces/
total 32
 0 drwxr-xr-x  5 jacquesmenu  staff   160 May 26 08:20:55 2021 ./
 0 drwxr-xr-x  8 jacquesmenu  staff   256 Jun 25 05:59:13 2021 ../
 8 -rw-r--r--@ 1 jacquesmenu  staff    77 Apr 22 15:49:27 2021 README.md
16 -rw-r--r--@ 1 jacquesmenu  staff  5796 Jun 24 17:47:02 2021 msrInterface.cpp
 8 -rw-r--r--@ 1 jacquesmenu  staff  1371 Jun 13 07:38:04 2021 msrInterface.h
```

34.2 LPSR interfaces

The LSPR interfaces are in `interfaces/lpsrinterfaces/`:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll interfaces/lpsrinterfaces/
total 24
0 drwxr-xr-x  5 jacquesmenu  staff   160 Jun 13 07:36:53 2021 ./
0 drwxr-xr-x  8 jacquesmenu  staff   256 Jun 25 05:59:13 2021 ../
8 -rw-r--r--@ 1 jacquesmenu  staff    78 Jun 13 07:37:13 2021 README.md
8 -rw-r--r--@ 1 jacquesmenu  staff   670 Jun 13 07:41:01 2021 lpsrInterface.cpp
8 -rw-r--r--@ 1 jacquesmenu  staff  1450 Jun 13 07:39:29 2021 lpsrInterface.h
```

34.3 MSDL interfaces

The MSDL interfaces are in `interfaces/msdlinterfaces/`:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > ll interfaces/msdlinterfaces/
total 8
0 drwxr-xr-x  3 jacquesmenu  staff    96 Jun 25 05:57:39 2021 ./
0 drwxr-xr-x  8 jacquesmenu  staff   256 Jun 25 05:59:13 2021 ../
8 -rw-r--r--@ 1 jacquesmenu  staff  1967 Jun  6 06:38:55 2021 libmsdl.h
```

Chapter 35

Passes interfaces

MusicFormats provides its functionality in two ways:

- a set of API functions providing its services to any application, including the ones hosted on the Web;
- a set of CLI services, to be used in terminals and scripts.

The CLI services actually use the API functions to do their job. For example, in , the `main ()` function does:

```
int main (int argc, char*  argv[])
//------------------------------------------------------------------------
{
  // setup signals catching
  // ------------------------------------------------------

  catchSignals ();

  // ... ... ...

  switch (multiGenerationOutputKind) {
    case mfMultiGenerationOutputKind::kGeneration_UNKNOWN_:
      // should not occur, unless the run is a pure help one
      break;

    // ... ... ...

    case mfMultiGenerationOutputKind::kGenerationGuido:
      err =
        msrScore2guidoWithHandler (
          theMsrScore,
          mfPassIDKind::kMfPassID_2,
          "Convert the MSR score into a second MSR",
          mfPassIDKind::kMfPassID_3,
          "Convert the second MSR into an MXSR",
          mfPassIDKind::kMfPassID_4,
          "Convert the MXSR into Guido text",
          std::cout,
          std::cerr,
          handler);
      break;

  // ... ... ...
}
```

35.1 Translating MusicXML data to an MXSR

35.2 Translating an MXSR to an MSR

35.3 Translating an MSR to an MXSR

35.4 Translating an MSR to another MSR

Such translation is meant to offer an opportunity to modify the score's description depending on options.

35.5 Translating an MSR to an LPSR

This converter embeds a specific converter of MSR to MSR, to circumvent the famous LilyPond issue #34.

35.6 Translating an LPSR to LilyPond code

35.7 Translating an MSR to an BSR

This converter embeds a specific converter of MSR to MSR, to circumvent the famous LilyPond issue #34.

35.8 Translating a BSR to another BSR

35.9 Translating an MXSR to Guido

Chapter 36

Converters interfaces

These interfaces are a set of functions to run the various converters. They are placed in the corresponding subdirectories of `src/converters/`, such as `src/converters/musicxml2musicxml/musicxml2musicxmlInterface.`

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/converters/musicxml2musicxml > cat
    musicxml2musicxmlInterface.h
/*
  MusicFormats Library
  Copyright (C) Jacques Menu 2016-2023

  This Source Code Form is subject to the terms of the Mozilla Public
  License, v. 2.0. If a copy of the MPL was not distributed with this
  file, you can obtain one at http://mozilla.org/MPL/2.0/.

  https://github.com/jacques-menu/musicformats
*/

#ifndef ___musicxml2musicxmlInterface___
#define ___musicxml2musicxmlInterface___

#include "mfMusicformatsErrors.h" // for mfMusicformatsErrorKind


namespace MusicFormats
{
/*
  The API functions with an options and arguments and no handler
  are declared in libmusicxml.h
*/

//_______________________________________________________________________________
EXP mfMusicformatsErrorKind convertMusicxmlFile2musicxmlWithHandler (
  const char*  fileName,
  std::ostream&     out,
  std::ostream&     err,
  const S_oahHandler& handler);

//_______________________________________________________________________________
EXP mfMusicformatsErrorKind convertMusicxmlFd2musicxmlWithHandler (
  FILE*        fd,
  std::ostream&     out,
  std::ostream&     err,
  const S_oahHandler& handler);

//_______________________________________________________________________________
EXP mfMusicformatsErrorKind convertMusicxmlString2musicxmlWithHandler (
  const char*  buffer,
  std::ostream&     out,
  std::ostream&     err,
  const S_oahHandler& handler);


}


#endif // ___musicxml2musicxmlInterface___
```

Part XII

Releases

36.1 The GitHub repository

The MusicFormats repository is hosted by GitHub and uses so-called *actions* to build the library on Mac OS™, Ubuntu and Windows™. The resulting files are then available in the repository, where they are available to create the releases for these three operating systems.

The releases Zip archives are supplied with all MusicFormats versions, i.e. the current, most recent version of MusicFormats (the default master branch in the repository), and the earlier versions such as the `v0.9.65` release.

The `musicformats-ubuntu-distrib` archive has to be clicked to get downloaded, since its URL cannot be guessed by an algorithm (it contains numbers internal to GitHub).

Doing so for the three releases, we get the following, here in the `Downloads/` folder on Mac OS™, with the Zip archives are automatically uncompressed :

```
jacquesmenu@macmini: ~/Downloads > ls -sal musicformats-*-distrib
musicformats-macos-distrib:
total 8
0 drwx------@  5 jacquesmenu  staff   160 Mar  3 09:18 .
0 drwx------+ 72 jacquesmenu  staff  2304 Mar  3 09:18 ..
8 -rw-r--r--@  1 jacquesmenu  staff     6 Mar  3 07:10 MusicFormatsVersionNumber.txt
0 drwxr-xr-x@  3 jacquesmenu  staff    96 Mar  3 09:18 build
0 drwxr-xr-x@  4 jacquesmenu  staff   128 Mar  3 09:18 documentation

musicformats-ubuntu-distrib:
total 8
0 drwx------@  5 jacquesmenu  staff   160 Mar  3 09:18 .
0 drwx------+ 72 jacquesmenu  staff  2304 Mar  3 09:18 ..
8 -rw-r--r--@  1 jacquesmenu  staff     6 Mar  3 07:31 MusicFormatsVersionNumber.txt
0 drwxr-xr-x@  4 jacquesmenu  staff   128 Mar  3 09:18 build
0 drwxr-xr-x@  4 jacquesmenu  staff   128 Mar  3 09:18 documentation

musicformats-windows-distrib:
total 8
0 drwx------@  5 jacquesmenu  staff   160 Mar  3 09:18 .
0 drwx------+ 72 jacquesmenu  staff  2304 Mar  3 09:18 ..
8 -rw-r--r--@  1 jacquesmenu  staff     6 Mar  3 07:43 MusicFormatsVersionNumber.txt
0 drwxr-xr-x@  4 jacquesmenu  staff   128 Mar  3 09:18 build
0 drwxr-xr-x@  4 jacquesmenu  staff   128 Mar  3 09:18 documentation
```

```
jacquesmenu@macmini: ~/Downloads > ls -sal musicformats-ubuntu-distrib/*
8 -rw-r--r--@ 1 jacquesmenu  staff  6 Mar  3 07:31 musicformats-ubuntu-distrib/
    MusicFormatsVersionNumber.txt

musicformats-ubuntu-distrib/build:
total 0
0 drwxr-xr-x@  4 jacquesmenu  staff  128 Mar  3 09:18 .
0 drwx------@  5 jacquesmenu  staff  160 Mar  3 09:18 ..
0 drwxr-xr-x@ 25 jacquesmenu  staff  800 Mar  3 09:18 bin
0 drwxr-xr-x@  4 jacquesmenu  staff  128 Mar  3 09:18 lib

musicformats-ubuntu-distrib/documentation:
total 0
0 drwxr-xr-x@ 4 jacquesmenu  staff  128 Mar  3 09:18 .
0 drwx------@ 5 jacquesmenu  staff  160 Mar  3 09:18 ..
0 drwxr-xr-x@ 3 jacquesmenu  staff   96 Mar  3 09:18 IntroductionToMusicXML
0 drwxr-xr-x@ 3 jacquesmenu  staff   96 Mar  3 09:18 mfuserguide
```

```
jacquesmenu@macmini: ~/Downloads > ls -sal musicformats-ubuntu-distrib/*/*
musicformats-ubuntu-distrib/build/bin:
total 2272
  0 drwxr-xr-x@ 25 jacquesmenu  staff     800 Mar  3 09:18 .
  0 drwxr-xr-x@  4 jacquesmenu  staff     128 Mar  3 09:18 ..
```

```
 96 -rw-r--r--@  1 jacquesmenu  staff   49008 Mar  3 07:31 LilyPondIssue34
 96 -rw-r--r--@  1 jacquesmenu  staff   49048 Mar  3 07:31 Mikrokosmos3Wandering
 96 -rw-r--r--@  1 jacquesmenu  staff   47280 Mar  3 07:31 MusicAndHarmonies
 96 -rw-r--r--@  1 jacquesmenu  staff   47272 Mar  3 07:31 RandomChords
 96 -rw-r--r--@  1 jacquesmenu  staff   47272 Mar  3 07:31 RandomMusic
 72 -rw-r--r--@  1 jacquesmenu  staff   33848 Mar  3 07:31 countnotes
 40 -rw-r--r--@  1 jacquesmenu  staff   17648 Mar  3 07:31 displayMusicformatsHistory
 40 -rw-r--r--@  1 jacquesmenu  staff   17648 Mar  3 07:31 displayMusicformatsVersion
104 -rw-r--r--@  1 jacquesmenu  staff   50400 Mar  3 07:31 msdlconverter
544 -rw-r--r--@  1 jacquesmenu  staff  276024 Mar  3 07:31 partsummary
 88 -rw-r--r--@  1 jacquesmenu  staff   43768 Mar  3 07:31 readunrolled
 80 -rw-r--r--@  1 jacquesmenu  staff   39064 Mar  3 07:31 xml2brl
 80 -rw-r--r--@  1 jacquesmenu  staff   39104 Mar  3 07:31 xml2gmn
 48 -rw-r--r--@  1 jacquesmenu  staff   23192 Mar  3 07:31 xml2guido
 72 -rw-r--r--@  1 jacquesmenu  staff   34816 Mar  3 07:31 xml2ly
 88 -rw-r--r--@  1 jacquesmenu  staff   42928 Mar  3 07:31 xml2midi
 80 -rw-r--r--@  1 jacquesmenu  staff   39104 Mar  3 07:31 xml2xml
 88 -rw-r--r--@  1 jacquesmenu  staff   43416 Mar  3 07:31 xmlclone
 48 -rw-r--r--@  1 jacquesmenu  staff   22616 Mar  3 07:31 xmlfactory
160 -rw-r--r--@  1 jacquesmenu  staff   79440 Mar  3 07:31 xmliter
 56 -rw-r--r--@  1 jacquesmenu  staff   28472 Mar  3 07:31 xmlread
 64 -rw-r--r--@  1 jacquesmenu  staff   28704 Mar  3 07:31 xmltranspose
 40 -rw-r--r--@  1 jacquesmenu  staff   17360 Mar  3 07:31 xmlversion

musicformats-ubuntu-distrib/build/lib:
total 158600
     0 drwxr-xr-x@ 4 jacquesmenu  staff       128 Mar  3 09:18 .
     0 drwxr-xr-x@ 4 jacquesmenu  staff       128 Mar  3 09:18 ..
113728 -rw-r--r--@ 1 jacquesmenu  staff  58227464 Mar  3 07:31 libmusicformats.a
 44872 -rw-r--r--@ 1 jacquesmenu  staff  22971160 Mar  3 07:31 libmusicformats.so

musicformats-ubuntu-distrib/documentation/IntroductionToMusicXML:
total 1704
   0 drwxr-xr-x@ 3 jacquesmenu  staff      96 Mar  3 09:18 .
   0 drwxr-xr-x@ 4 jacquesmenu  staff     128 Mar  3 09:18 ..
1704 -rw-r--r--@ 1 jacquesmenu  staff  869211 Mar  3 07:31 IntroductionToMusicXML.pdf

musicformats-ubuntu-distrib/documentation/mfuserguide:
total 3000
   0 drwxr-xr-x@ 3 jacquesmenu  staff       96 Mar  3 09:18 .
   0 drwxr-xr-x@ 4 jacquesmenu  staff      128 Mar  3 09:18 ..
3000 -rw-r--r--@ 1 jacquesmenu  staff  1532300 Mar  3 07:31 mfuserguide.pdf
```

The contents of `musicformats-windows-distrib/` differs in the `lib/` contents:

```
jacquesmenu@macmini: ~/Downloads > ls -sal musicformats-windows-distrib/build/lib/
total 37672
    0 drwxr-xr-x@ 4 jacquesmenu  staff       128 Mar  3 09:18 .
    0 drwxr-xr-x@ 4 jacquesmenu  staff       128 Mar  3 09:18 ..
14768 -rw-r--r--@ 1 jacquesmenu  staff   7558913 Mar  3 07:44 musicformats.exp
22904 -rw-r--r--@ 1 jacquesmenu  staff  11726392 Mar  3 07:44 musicformats.lib
```

For Mac OS™, there is no `lib/` directory, since the executables in `bin/` are self-sufficient. They can be placed anywhere on a disk except the trash. Usually, they are placed in the `/Applications/` directory.

36.1.1 Creating the releases

The hierarchy in the `musicformats-*-distrib/` directories comes from the MusicFormats repository untouched, which is not convenient for the users.

Their contents is thus re-structured by `devtools/MakeMusicFormatsDistributions.bash`:

```
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/MakeMusicFormatsDistributions.bash
... ... ...

==> final distrib contents:
----------------------------------------------

  4208 -rw-r--r--  1 jacquesmenu  staff   2153547 Mar  3 12:55 /Users/jacquesmenu/
    musicformats-git-dev/distrib/MusicFormatsForWindows.zip
 34576 -rw-r--r--  1 jacquesmenu  staff  17559638 Mar  3 12:55 /Users/jacquesmenu/
    musicformats-git-dev/distrib/MusicFormatsForUbuntu.zip
109512 -rw-r--r--  1 jacquesmenu  staff  55888914 Mar  3 12:55 /Users/jacquesmenu/
    musicformats-git-dev/distrib/MusicFormatsForMacOS.zip
  1704 -rw-r--r--@ 1 jacquesmenu  staff    869211 Mar  3 07:10 /Users/jacquesmenu/
    musicformats-git-dev/distrib/IntroductionToMusicXML.pdf
  3000 -rw-r--r--@ 1 jacquesmenu  staff   1532300 Mar  3 07:10 /Users/jacquesmenu/
    musicformats-git-dev/distrib/mfuserguide.pdf
     8 -rw-r--r--@ 1 jacquesmenu  staff         6 Mar  3 07:10 /Users/jacquesmenu/
    musicformats-git-dev/distrib/MusicFormatsVersionNumber.txt

/Users/jacquesmenu/musicformats-git-dev/distrib/MusicFormatsForWindows:
total 8
0 drwxr-xr-x  16 jacquesmenu  staff  512 Mar  3 12:55 ..
0 drwxr-xr-x   5 jacquesmenu  staff  160 Mar  3 12:55 .
0 drwxr-xr-x@  4 jacquesmenu  staff  128 Mar  3 09:18 lib
0 drwxr-xr-x@ 25 jacquesmenu  staff  800 Mar  3 09:18 bin
8 -rw-r--r--@  1 jacquesmenu  staff    6 Mar  3 07:43 MusicFormatsVersionNumber.txt

/Users/jacquesmenu/musicformats-git-dev/distrib/MusicFormatsForUbuntu:
total 8
0 drwxr-xr-x  16 jacquesmenu  staff  512 Mar  3 12:55 ..
0 drwxr-xr-x   5 jacquesmenu  staff  160 Mar  3 12:55 .
0 drwxr-xr-x@  4 jacquesmenu  staff  128 Mar  3 09:18 lib
0 drwxr-xr-x@ 25 jacquesmenu  staff  800 Mar  3 09:18 bin
8 -rw-r--r--@  1 jacquesmenu  staff    6 Mar  3 07:31 MusicFormatsVersionNumber.txt

/Users/jacquesmenu/musicformats-git-dev/distrib/MusicFormatsForMacOS:
total 8
0 drwxr-xr-x  16 jacquesmenu  staff  512 Mar  3 12:55 ..
0 drwxr-xr-x   4 jacquesmenu  staff  128 Mar  3 12:55 .
0 drwxr-xr-x@ 25 jacquesmenu  staff  800 Mar  3 07:10 bin
8 -rw-r--r--@  1 jacquesmenu  staff    6 Mar  3 07:10 MusicFormatsVersionNumber.txt
```

The contents of distrib/ is now:

```
jacquesmenu@macmini: ~/musicformats-git-dev/distrib > ls -sal
total 154128
     0 drwxr-xr-x  16 jacquesmenu  staff       512 Mar  3 13:18 .
     0 drwxr-xr-x  35 jacquesmenu  staff      1120 Mar  3 07:13 ..
    24 -rw-r--r--@  1 jacquesmenu  staff      8196 Feb 24 13:33 .DS_Store
  1704 -rw-r--r--@  1 jacquesmenu  staff    869211 Mar  3 07:10 IntroductionToMusicXML.pdf
     0 drwxr-xr-x   4 jacquesmenu  staff       128 Mar  3 13:18 MusicFormatsForMacOS
109960 -rw-r--r--   1 jacquesmenu  staff  55888914 Mar  3 13:18 MusicFormatsForMacOS.zip
     0 drwxr-xr-x   5 jacquesmenu  staff       160 Mar  3 13:18 MusicFormatsForUbuntu
 35216 -rw-r--r--   1 jacquesmenu  staff  17559638 Mar  3 13:18 MusicFormatsForUbuntu.zip
     0 drwxr-xr-x   5 jacquesmenu  staff       160 Mar  3 13:18 MusicFormatsForWindows
  4208 -rw-r--r--   1 jacquesmenu  staff   2153547 Mar  3 13:18 MusicFormatsForWindows.zip
  3000 -rw-r--r--@  1 jacquesmenu  staff   1532300 Mar  3 07:10 mfuserguide.pdf
     8 -rw-r--r--@  1 jacquesmenu  staff         6 Mar  3 07:10 MusicFormatsVersionNumber.
    txt
     8 -rwxr-xr-x@  1 jacquesmenu  staff        95 Mar  3 12:54 doClean.bash
     0 drwx------@  6 jacquesmenu  staff       192 Mar  3 10:56 musicformats-macos-distrib
     0 drwx------@  6 jacquesmenu  staff       192 Mar  3 10:56 musicformats-ubuntu-
    distrib
     0 drwx------@  6 jacquesmenu  staff       192 Mar  3 10:56 musicformats-windows-
    distrib
```

36.1.2 Security issue in recent MacOS™ versions

Mac OS™ gets more and more stringent over time regarding security. The operating system part in charge of this is named Gatekeeper.

When downloading the MusicFormats releases from the repository on versions up to 10 (High Sierra), the executables in `bin` are usable alright.

From version 11 (Catalina) on, though, the executables you get are not executable actually, because their developer is unknown to the operating system, and actions have to be taken for them to be usable.

The trouble is that these executables are in *quarantine* by default. To make them usable, they have to quit quarantine and explicitly be made executable.

This is done this way using `chmod` and `xattr` in `devtools/MakeMusicFormatsDistributions.bash`:

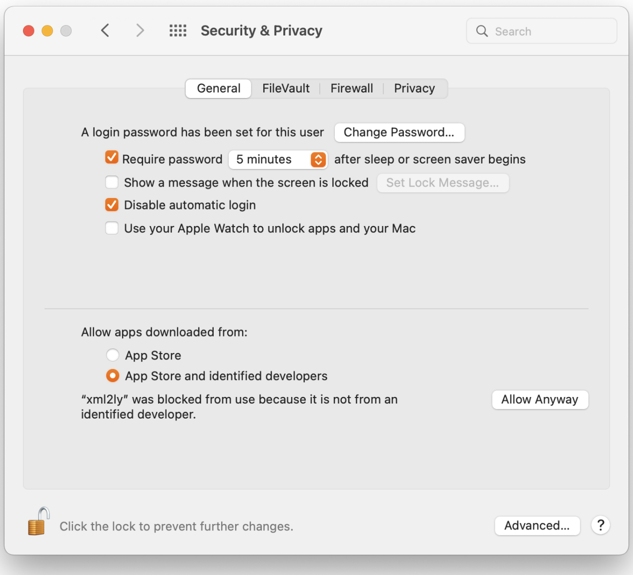
```
# make the executables actually executable
chmod +x bin/*
xattr -d com.apple.quarantine bin/*
```

From then on, the MusicFormats executables can be used seamlessly on the given machine.

Having to perform the preceding task for each executable is the price to pay for security. And it has to be perfomed again when installing new versions...

The above can be done in the GUI file by file too. Right after you got the message above:

- open *System Preferences*, choose the *Security & Privacy* tab, and there click on the *General button*;
- click on the lock at the bottom left of the dialog to make changes:



A login password has been set for this user

✓ Require password 5 minutes after sleep or screen saver begins

☐ Show a message when the screen is locked

✓ Disable automatic login

☐ Use your Apple Watch to unlock apps and your Mac

Allow apps downloaded from:

○ App Store

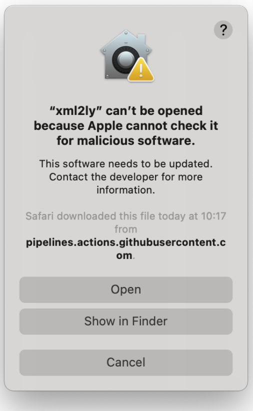
● App Store and identified developers

"xml2ly" was blocked from use because it is not from an identified developer.

Click the lock to prevent further changes.

- click on the *Allow Anyway* button.

Re-execute the executable from the command line. This pops-up a dialog to confirm you actually want to use this software:


"xml2ly" can't be opened because Apple cannot check it for malicious software.

This software needs to be updated. Contact the developer for more information.

Safari downloaded this file today at 10:17 from pipelines.actions.githubusercontent.com.

Click on the *Open* button to register the executable in Gatekeeper and go ahead.

36.2 GitHub actions

These actions are defined in `.yml` files in `.github/workflows/`:

```
jacquesmenu@macmini:~/musicformats-git-dev/.github/workflows > ls -sal
total 24
0 drwxr-xr-x  5 jacquesmenu  staff   160 Aug 24 09:35 .
0 drwxr-xr-x  4 jacquesmenu  staff   128 Aug 22 08:41 ..
8 -rw-r--r--@ 1 jacquesmenu  staff  1366 Aug 23 07:09 build-macos-version.yml
8 -rw-r--r--@ 1 jacquesmenu  staff  1371 Aug 23 07:09 build-ubuntu-version.yml
8 -rw-r--r--@ 1 jacquesmenu  staff  1455 Aug 23 07:08 build-windows-version.yml
```

These actions are executed by default each time a `git push` is performed to the master branch. This can be disabled at will if needed.

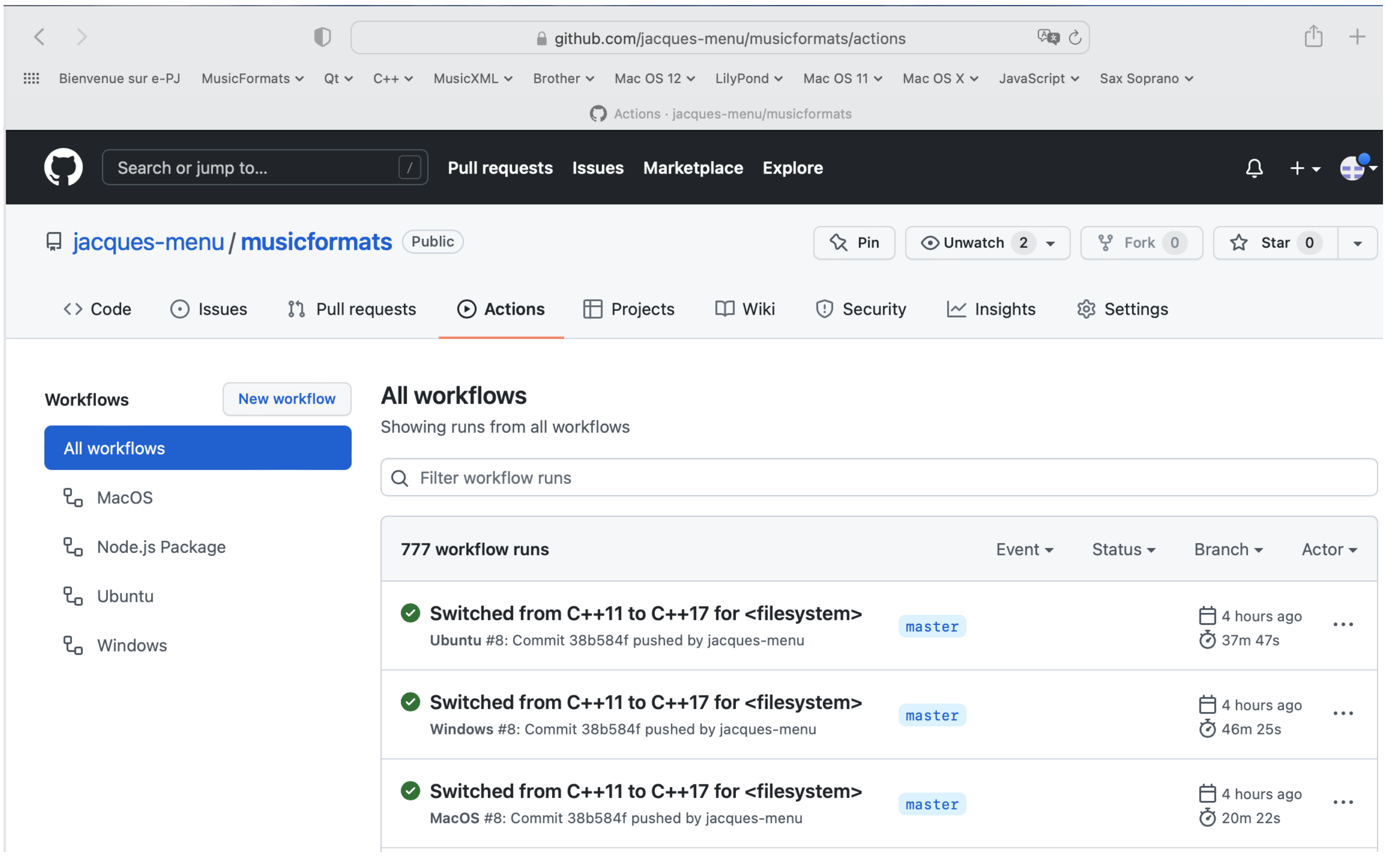
For example, the Ubuntu action in file `build-ubuntu-version.yml` is shown below:

```
jacquesmenu@macstudio-1:/Volumes/JMI_Volume/JMI_Developpement/musicformats-git-dev/.github
    /workflows > cat build-ubuntu-artifact.yml
# This is a workflow to build MusicFormats and create an artifact of it

name: Build Ubuntu Version

# Controls when the action will run.
on:
  # Triggers the workflow on push or pull request events but only for the master branch
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest

    # Steps represent a sequence of tasks that will be executed as part of the job
    steps:
      # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '16'

      - name: Build MusicFormats for Ubuntu
        run: make -C build

      - name: Upload libraries and executables for Ubuntu
        uses: actions/upload-artifact@v3
        with:
          name: musicformats-ubuntu-artifact
          path: |
            MusicFormatsVersionNumber.txt
            MusicFormatsVersionDate.txt
            build/bin
            build/lib
            include
            documentation/IntroductionToMusicXML/IntroductionToMusicXML.pdf
            documentation/mfuserguide/mfuserguide.pdf
            documentation/MusicFormatsFigures/MusicFormatsFigures.pdf
            documentation/mfmaintainanceguide/mfmaintainanceguide.pdf
            #uncomment to have the full libdir in the .zip archive
            #build/libdir
```

After a push to the master branch:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git commit -m "Switched from C++11 to C++17
    for <filesystem>" -a[master 77d3d29] Switched from C++11 to C++17 for <filesystem>
 7 files changed, 10 insertions(+), 10 deletions(-)

jacquesmenu@macmini: ~/musicformats-git-dev > git push
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (17/17), 1.47 KiB | 1.47 MiB/s, done.
Total 17 (delta 14), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (14/14), completed with 13 local objects.
To https://github.com/jacques-menu/musicformats.git
   a880063..77d3d29  master -> master
```

we get for example:



All workflows

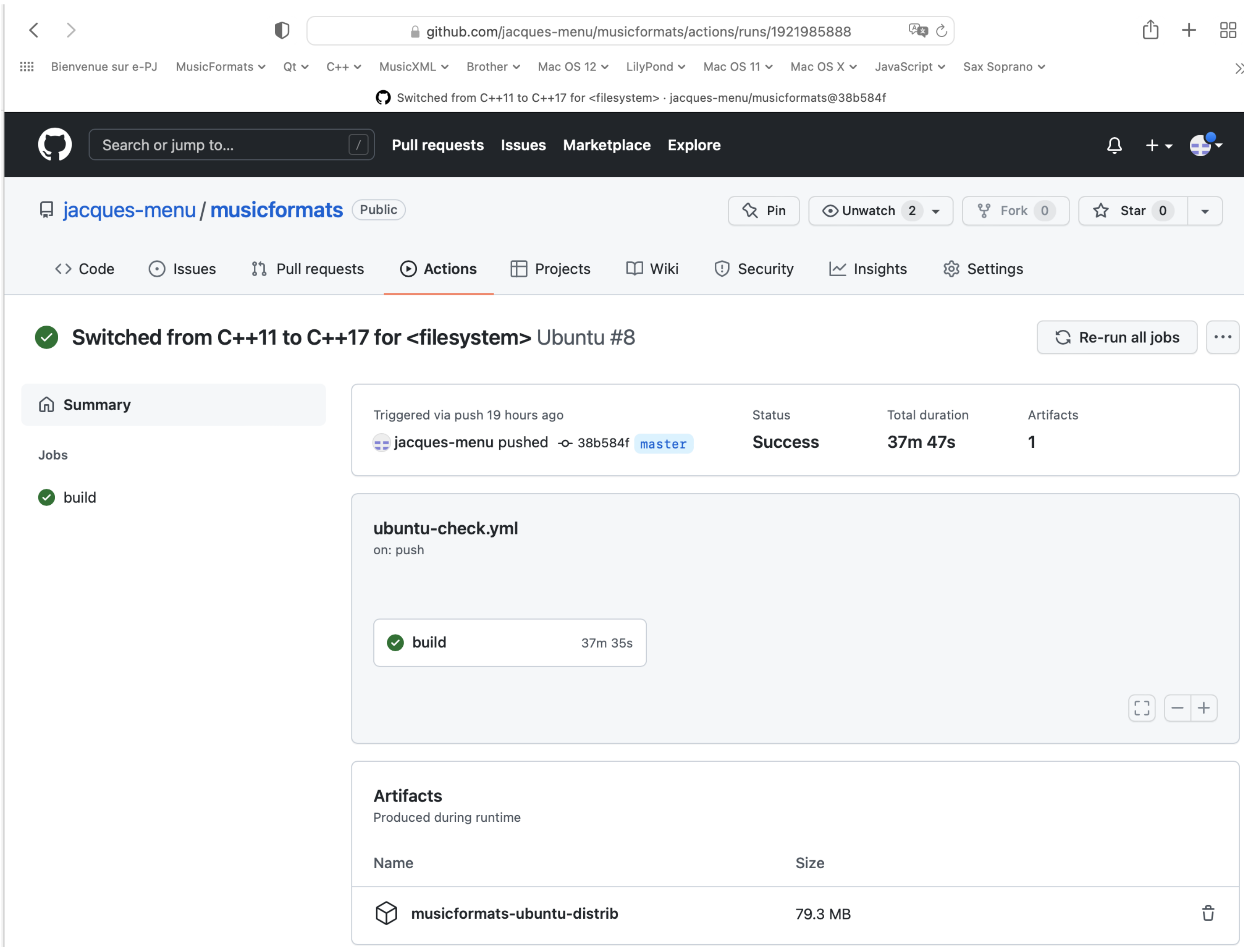
Showing runs from all workflows

Filter workflow runs

777 workflow runs

Switched from C++11 to C++17 for <filesystem> **Ubuntu** #8: Commit 38b584f pushed by jacques-menu	master	4 hours ago 37m 47s	...
Switched from C++11 to C++17 for <filesystem> **Windows** #8: Commit 38b584f pushed by jacques-menu	master	4 hours ago 46m 25s	...
Switched from C++11 to C++17 for <filesystem> **MacOS** #8: Commit 38b584f pushed by jacques-menu	master	4 hours ago 20m 22s	...

Then cliking on the link leads to:



Switched from C++11 to C++17 for <filesystem> · jacques-menu/musicformats@38b584f

jacques-menu / musicformats Public

Switched from C++11 to C++17 for <filesystem> Ubuntu #8

Triggered via push 19 hours ago
jacques-menu pushed 38b584f master

Status
Success

Total duration
37m 47s

Artifacts
1

ubuntu-check.yml
on: push

build 37m 35s

Artifacts
Produced during runtime

Name	Size
musicformats-ubuntu-distrib	79.3 MB

Chapter 37

MusicFormats releases

The MusicFormats repository contains a single, master branch, that contains the current evolution of the code base, examples and documentation.

When a `git push` is performed, the `musicformats-*-artifact` archives are created, but they cannot be added to the MusicFormats repository by GitHubbecause they are too large.

In order to create a new release, one should:

1. (re-)create the up-to-date documentation with:

```
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/CreateDocumentationPDFs.bash
```

2. add all new and/or modified files to the local repository. The `addAll` function is defined for this:

```
jacquesmenu@macstudio:/Volumes/JMI_Volume/JMI_Developpement/musicformats-git-dev >
    typ addAll
addAll is a shell function from /Volumes/JMI_VOLUME/JMI_ShellSettings/
    zsh_20_MusicFormats
--
addAll () {
  git add -f ${MUSIC_FORMATS_DEV}/MusicFormatsVersionNumber.txt
  git add -f ${MUSIC_FORMATS_DEV}/MusicFormatsVersionDate.txt
  git add -f ${MUSIC_FORMATS_DEV}/src/MusicFormatsVersionNumber.h
  git add -f ${MUSIC_FORMATS_DEV}/src/MusicFormatsVersionDate.h
  addSrc
  addBuild
  addDevtools
  addInclude
  addDoc
  addFxml
  addFmfsl
}
```

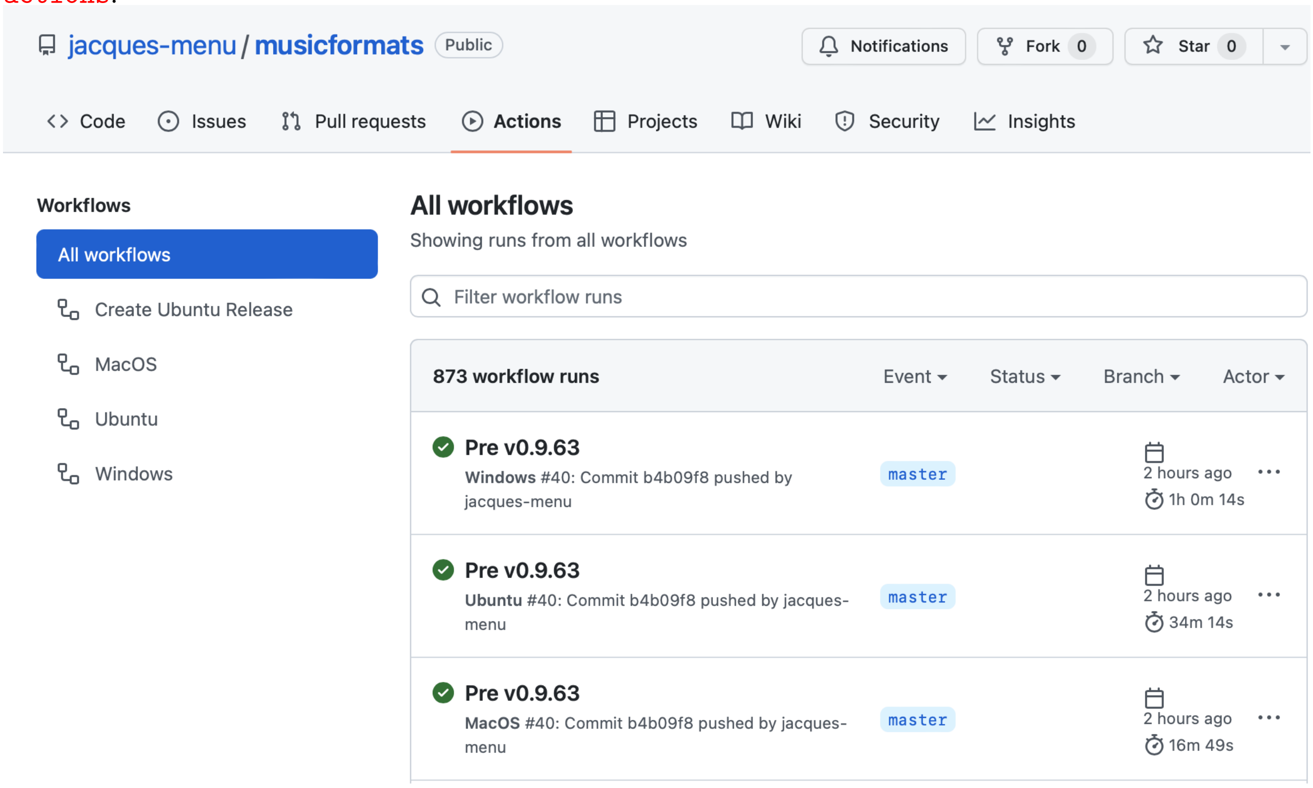
3. commit to the local repository clone with the release version number:

```
git commit -m "v0.9.67-alpha" -a
```

4. push this to the MusicFormats repo with:

```
git push
```

5. the actions in the MusicFormats repository then perform a build on Linux, Windows™and Mac OS™. Check that they were executed successfully at https://github.com/jacques-menu/musicformats/actions:



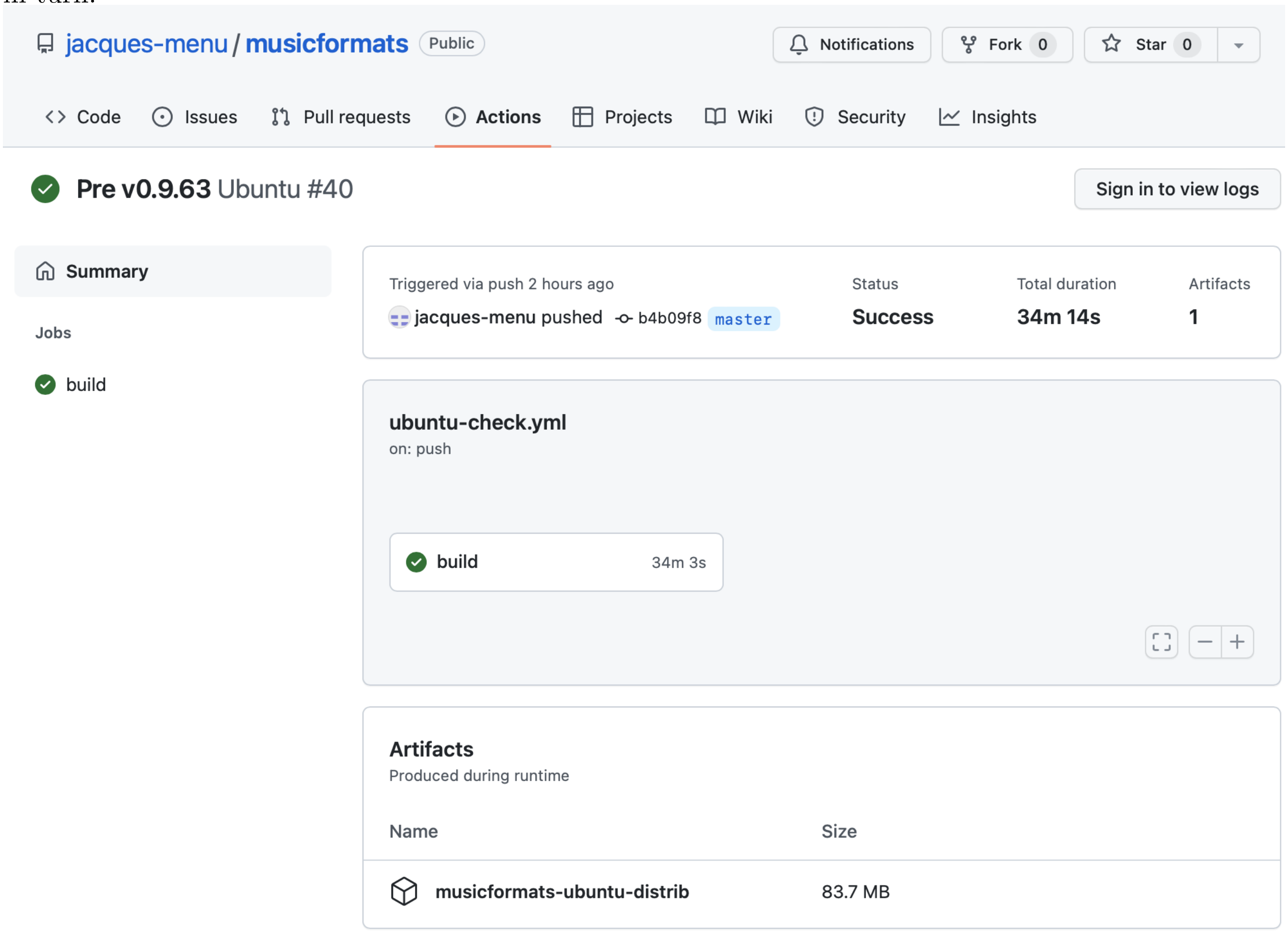
All workflows

Showing runs from all workflows

873 workflow runs

Pre v0.9.63 Windows #40: Commit b4b09f8 pushed by jacques-menu	master	2 hours ago 1h 0m 14s
Pre v0.9.63 Ubuntu #40: Commit b4b09f8 pushed by jacques-menu	master	2 hours ago 34m 14s
Pre v0.9.63 MacOS #40: Commit b4b09f8 pushed by jacques-menu	master	2 hours ago 16m 49s

6. when that is the case, download each of the three resulting `musicformats-*-distrib` archives locally in turn:



Pre v0.9.63 Ubuntu #40

Triggered via push 2 hours ago	Status	Total duration	Artifacts
jacques-menu pushed b4b09f8 master	Success	34m 14s	1

ubuntu-check.yml
on: push

build 34m 3s

Artifacts
Produced during runtime

Name	Size
musicformats-ubuntu-distrib	83.7 MB

On this authors's machine, they go to ${HOME}/Downloads:

```
jacquesmenu@macstudio:~/JMI_Downloads > ls -sal musicformats-*-artifact
musicformats-macos-artifact:
total 32
 0 drwx------@  8 jacquesmenu  staff   256 Mar 13 14:26 .
 0 drwxr-xr-x@ 65 jacquesmenu  staff  2080 Mar 13 14:29 ..
16 -rw-r--r--@  1 jacquesmenu  staff  6148 Mar 13 14:27 .DS_Store
 8 -rw-r--r--@  1 jacquesmenu  staff    15 Mar 13 12:49 MusicFormatsVersionDate.txt
 8 -rw-r--r--@  1 jacquesmenu  staff    12 Mar 13 12:49 MusicFormatsVersionNumber.txt
 0 drwxr-xr-x@  5 jacquesmenu  staff   160 Mar 13 14:26 build
 0 drwxr-xr-x@  7 jacquesmenu  staff   224 Mar 13 14:26 documentation
 0 drwxr-xr-x@ 18 jacquesmenu  staff   576 Mar 13 14:26 include

musicformats-ubuntu-artifact:
total 32
 0 drwx------@  8 jacquesmenu  staff   256 Mar 13 14:26 .
 0 drwxr-xr-x@ 65 jacquesmenu  staff  2080 Mar 13 14:29 ..
16 -rw-r--r--@  1 jacquesmenu  staff  6148 Mar 13 14:27 .DS_Store
 8 -rw-r--r--@  1 jacquesmenu  staff    15 Mar 13 12:41 MusicFormatsVersionDate.txt
 8 -rw-r--r--@  1 jacquesmenu  staff    12 Mar 13 12:41 MusicFormatsVersionNumber.txt
 0 drwxr-xr-x@  5 jacquesmenu  staff   160 Mar 13 14:26 build
 0 drwxr-xr-x@  7 jacquesmenu  staff   224 Mar 13 14:26 documentation
 0 drwxr-xr-x@ 18 jacquesmenu  staff   576 Mar 13 14:26 include

musicformats-windows-artifact:
total 32
 0 drwx------@  8 jacquesmenu  staff   256 Mar 13 14:26 .
 0 drwxr-xr-x@ 65 jacquesmenu  staff  2080 Mar 13 14:29 ..
16 -rw-r--r--@  1 jacquesmenu  staff  6148 Mar 13 14:27 .DS_Store
 8 -rw-r--r--@  1 jacquesmenu  staff    16 Mar 13 12:59 MusicFormatsVersionDate.txt
 8 -rw-r--r--@  1 jacquesmenu  staff    12 Mar 13 12:59 MusicFormatsVersionNumber.txt
 0 drwxr-xr-x@  5 jacquesmenu  staff   160 Mar 13 14:26 build
 0 drwxr-xr-x@  7 jacquesmenu  staff   224 Mar 13 14:26 documentation
 0 drwxr-xr-x@ 18 jacquesmenu  staff   576 Mar 13 14:26 include
```

7. create the releases in the local MusicFormats repository clone:

```
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/MakeMusicFormatsDistributions.
    bash
... ... ...
----------------------------------------------
==> final distrib contents:
----------------------------------------------

  4368 -rw-r--r--  1 jacquesmenu  staff   2234811 Jun  9 11:47 /Users/jacquesmenu/
    musicformats-git-dev/releases/MusicFormatsForWindows.zip
 35312 -rw-r--r--  1 jacquesmenu  staff  18076799 Jun  9 11:47 /Users/jacquesmenu/
    musicformats-git-dev/releases/MusicFormatsForUbuntu.zip
127824 -rw-r--r--  1 jacquesmenu  staff  65442854 Jun  9 11:47 /Users/jacquesmenu/
    musicformats-git-dev/releases/MusicFormatsForMacOS.zip
  1712 -rw-r--r--@ 1 jacquesmenu  staff    872863 Jun  9 07:40 /Users/jacquesmenu/
    musicformats-git-dev/releases/IntroductionToMusicXML.pdf
  5328 -rw-r--r--@ 1 jacquesmenu  staff   2724130 Jun  9 07:40 /Users/jacquesmenu/
    musicformats-git-dev/releases/mfuserguide.pdf
     8 -rw-r--r--@ 1 jacquesmenu  staff         6 Jun  9 07:40 /Users/jacquesmenu/
    musicformats-git-dev/releases/MusicFormatsVersionNumber.txt

/Users/jacquesmenu/musicformats-git-dev/releases/MusicFormatsForWindows:
total 8
0 drwxr-xr-x  14 jacquesmenu  staff  448  Jun  9 11:47 ..
0 drwxr-xr-x   5 jacquesmenu  staff  160  Jun  9 11:47 .
0 drwxr-xr-x@  4 jacquesmenu  staff  128  Jun  9 11:43 lib
0 drwxr-xr-x@ 26 jacquesmenu  staff  832  Jun  9 11:43 bin
8 -rw-r--r--@  1 jacquesmenu  staff    6  Jun  9 08:14 MusicFormatsVersionNumber.txt

/Users/jacquesmenu/musicformats-git-dev/releases/MusicFormatsForUbuntu:
```

```
total 8
0 drwxr-xr-x  14 jacquesmenu  staff  448 Jun  9 11:47 ..
0 drwxr-xr-x   5 jacquesmenu  staff  160 Jun  9 11:47 .
0 drwxr-xr-x@  4 jacquesmenu  staff  128 Jun  9 11:44 lib
0 drwxr-xr-x@ 26 jacquesmenu  staff  832 Jun  9 11:44 bin
8 -rw-r--r--@  1 jacquesmenu  staff    6 Jun  9 07:57 MusicFormatsVersionNumber.txt

/Users/jacquesmenu/musicformats-git-dev/releases/MusicFormatsForMacOS:
total 8
0 drwxr-xr-x  14 jacquesmenu  staff  448 Jun  9 11:47 ..
0 drwxr-xr-x   4 jacquesmenu  staff  128 Jun  9 11:47 .
0 drwxr-xr-x@ 26 jacquesmenu  staff  832 Jun  9 07:40 bin
8 -rw-r--r--@  1 jacquesmenu  staff    6 Jun  9 07:40 MusicFormatsVersionNumber.txt
```

Now, the `releases` folder contains the MusicFormats new releases for the three operating systems:

```
jacquesmenu@macmini: ~/musicformats-git-dev/releases > ls -al
total 174552
drwxr-xr-x  14 jacquesmenu  staff       448 Jun  9 11:47 .
drwxr-xr-x  29 jacquesmenu  staff       928 Jun  9 09:13 ..
-rw-r--r--@  1 jacquesmenu  staff    872863 Jun  9 07:40 IntroductionToMusicXML.pdf
drwxr-xr-x   4 jacquesmenu  staff       128 Jun  9 11:47 MusicFormatsForMacOS
-rw-r--r--   1 jacquesmenu  staff  65442854 Jun  9 11:47 MusicFormatsForMacOS.zip
drwxr-xr-x   5 jacquesmenu  staff       160 Jun  9 11:47 MusicFormatsForUbuntu
-rw-r--r--   1 jacquesmenu  staff  18076799 Jun  9 11:47 MusicFormatsForUbuntu.zip
drwxr-xr-x   5 jacquesmenu  staff       160 Jun  9 11:47 MusicFormatsForWindows
-rw-r--r--   1 jacquesmenu  staff   2234811 Jun  9 11:47 MusicFormatsForWindows.zip
-rw-r--r--@  1 jacquesmenu  staff   2724130 Jun  9 07:40 mfuserguide.pdf
-rw-r--r--@  1 jacquesmenu  staff         6 Jun  9 07:40 MusicFormatsVersionNumber.
    txt
drwx------@  5 jacquesmenu  staff       160 Jun  9 11:44 musicformats-macos-distrib
drwx------@  5 jacquesmenu  staff       160 Jun  9 11:44 musicformats-ubuntu-distrib
drwx------@  5 jacquesmenu  staff       160 Jun  9 11:43 musicformats-windows-distrib
```

8. create the new release repository on the remote GitHub repository:
   (a) click on the Releases link in the repository root window's right pane;
   (b) click on the 'Draft a new release' button;
   (c) clik on the 'Choose a tag' button to create a new tag such as 'v0.9.67-alpha';
   (d) describe this release as needed;
   (e) drop the three Zip files in the `releases` folder to attach them to this release;
   (f) click on the 'Publish release' button to copy the Zip file to GitHub.
9. create a new version number and date, for example:

```
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/SetMusicFormatsVersionNumber.
    bash 0.9.64
-bash: devtools/SetMusicFormatsVersionNumber.: No such file or directory
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/SetMusicFormatsVersionNumber.
    bash 0.9.64
==> PWD is:
/Users/jacquesmenu/musicformats-git-dev

==> Writing MusicFormats version number 0.9.64 to MusicFormatsVersionNumber.txt

8 -rw-r--r--  1 jacquesmenu  staff  6 Jun  9 12:14:57 2022 MusicFormatsVersionNumber.
    txt
0.9.64
==> PWD is:
/Users/jacquesmenu/musicformats-git-dev/src

==> Writing MusicFormats version number 0.9.64 to MusicFormatsVersionNumber.h

```

```
8 -rw-r--r--  1 jacquesmenu  staff  45 Jun  9 12:14:57 2022 MusicFormatsVersionNumber
    .h
#define MUSICFORMATS_VERSION_NUMBER "0.9.64"
```

```
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/SetMusicFormatsVersionDate.
    bash "June 9, 2022"
==> PWD is:
/Users/jacquesmenu/musicformats-git-dev

==> Writing MusicFormats version date June 9, 2022 to MusicFormatsVersionDate.txt

8 -rw-r--r--  1 jacquesmenu  staff  13 Jun  9 12:15:52 2022 MusicFormatsVersionDate.
    txt
June 9, 2022

==> PWD is:
/Users/jacquesmenu/musicformats-git-dev/src

==> Writing MusicFormats version date June 9, 2022 to MusicFormatsVersionDate.h

8 -rw-r--r--  1 jacquesmenu  staff  49 Jun  9 12:15:52 2022 MusicFormatsVersionDate.h
#define MUSICFORMATS_VERSION_DATE "June 9, 2022"
```

Check the result with:

```
jacquesmenu@macmini: ~/musicformats-git-dev > devtools/ShowMusicFormatsVersion.bash
Version number:
0.9.64Version date:
June 9, 2022
```

Part XIII

Selected topics

Chapter 38

Initializations

Some initialization activities in MusicFormats use the OAH facility. OAH should thus be initialized first.

38.1 Options and help initializations

There is no initialization of the OAH architecture as such, but there are `create*OahGroup ()` functions to create the various OAH groups.

For example, global variable `gServiceRunData` is supplied by `src/mflibrary/mfServiceRunData.h/.cpp`:

```
EXP extern S_generalOahGroup gServiceRunData;

//______________________________________________________________________________
EXP S_generalOahGroup createGlobalGeneralOahGroup ();
```

```
S_generalOahGroup createGlobalGeneralOahGroup ()
{
#ifdef MF_TRACE_IS_ENABLED
  if (gEarlyOptions.getTraceEarlyOptions ()) {
    std::stringstream ss;

    ss <<
      "Creating global general OAH group";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // protect library against multiple initializations
  if (! gServiceRunData) {
    // create the global general options group
    gServiceRunData =
      generalOahGroup::create ();
    assert (gServiceRunData != 0);
  }

  // return the global OAH group
  return gServiceRunData;
}
```

38.2 Representations initializations

There are **initialize* ()** functions such as **initializeLPSR ()** and **initializeBSR ()**. They essentially build global data structures, such as the tables of supported languages and their correspondance with an internal enumeration type both ways.

For example, **initializeMSR ()** is defined in src/formats/msr/msr.h/.cpp:

```
void EXP initializeMSR ();
```

```
static S_mfcVersions pMsrRepresentationComponent;

static void initializeMsrRepresentationComponent ()
{
  // create the component
  pMsrRepresentationComponent =
    mfcVersions::create ("MSR");

  // populate it
  pMsrRepresentationComponent->
    appendVersionDescrToComponent (
      mfcVersionDescr::create (
        mfcVersionDescr::create (
          mfcVersionNumber::createFromString ("0.9.50"),
          "October 6, 2021",
          std::list<std::string> {
            "Start of sequential versions numbering"
          }
      )));
}

void initializeMSR ()
{
  // protect library against multiple initializations
  static Bool pPrivateThisMethodHasBeenRun (false);

  if (! pPrivateThisMethodHasBeenRun) {
    // initialize the history
    initializeMsrRepresentationComponent ();

    // initialize
    initializeMsrBasicTypes ();

    pPrivateThisMethodHasBeenRun = true;
  }
}
```

38.2.1 MSR initialization

src/formats/msr/msrBasicTypes.h/.cpp defines function **initializeMSRBasicTypes ()** for this initialization:

```
void initializeMsrBasicTypes ()
{
  // protect library against multiple initializations
  static Bool pPrivateThisMethodHasBeenRun (false);

  if (! pPrivateThisMethodHasBeenRun) {
#ifdef MF_TRACE_IS_ENABLED
    if (gEarlyOptions.getTraceEarlyOptions () && ! gEarlyOptions.getEarlyQuietOption ()) {
      gLog <<
        "Initializing MSR basic types handling" <<
        std::endl;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

    // languages handling
    // ------------------------------------------------------

    initializeQuarterTonesPitchesLanguageKinds ();

    // clefs handling
    // ------------------------------------------------------

    initializeClefKinds ();

    // harmonies handling
    // ------------------------------------------------------

    initializeHarmonyKinds ();

    // harmony structures handling
    // ------------------------------------------------------

    initializeHarmonyStructuresMap ();

    // MSR lengths handling
    // ------------------------------------------------------

    initializeMsrLengthUnitKindsMap ();

    // MSR margins types handling
    // ------------------------------------------------------

    initializeMsrMarginTypeKindsMap ();

    pPrivateThisMethodHasBeenRun = true;
  }
}
```

38.2.2 LPSR initialization

38.2.3 BSR initialization

38.3 Passes initializations

38.4 Converters initializations

The converters create only the global OAH groups they need. Since the order of initializations is critical, initialization of the formats is done when the latter's insider handler is created.

This is how class `xml2lyInsiderHandler` initializes the MSR and LSPR formats in method `xml2lyInsiderHandler::createTheXml2lyOptionGroups ()` in `src/converters/musicxml2lilypond/musicxml2lilypondInsiderHandler.cpp`:

```
void xml2lyInsiderHandler::createTheXml2lyOptionGroups (
  std::string serviceName)
{
  // ... ... ...

  // initialize options handling, phase 1
  // ------------------------------------------------------

  // create the OAH OAH group first
  appendGroupToHandler (
    createGlobalOahOahGroup (
      serviceName));

  // create the WAE OAH group
  appendGroupToHandler (
    createGlobalWaeOahGroup ());

#ifdef MF_TRACE_IS_ENABLED
  // create the trace OAH group
  appendGroupToHandler (
    createGlobalTraceOahGroup (
      this));
#endif // MF_TRACE_IS_ENABLED

  // create the output file OAH group
  appendGroupToHandler (
    createGlobalOutputFileOahGroup ());

  // initialize the library
  // ------------------------------------------------------

  initializeWAE ();

  initializeMSR ();
  initializeLPSR ();

  // initialize options handling, phase 2
  // ------------------------------------------------------

  // create the MXSR OAH group
  appendGroupToHandler (
    createGlobalMxsrOahGroup ());

  // create the mxsr2msr OAH group
  appendGroupToHandler (
    createGlobalMxsr2msrOahGroup (
      this));

  // create the MSR OAH group
  appendGroupToHandler (
    createGlobalMsrOahGroup ());

  // create the msr2msr OAH group
  appendGroupToHandler (
    createGlobalMsr2msrOahGroup ());

  // create the msr2lpsr OAH group
  appendGroupToHandler (
    createGlobalMsr2lpsrOahGroup ());

  // create the LPSR OAH group
  appendGroupToHandler (
    createGlobalLpsrOahGroup ());

  // create the LilyPond generation OAH group
  appendGroupToHandler (
    createGlobalLpsr2lilypondOahGroup ());

  // create the extra OAH group
  appendGroupToHandler (
    createGlobalHarmoniesExtraOahGroup ());

  // create the global xml2ly OAH group only now,
  // after the groups whose options it may use
  // have been created
  appendGroupToHandler (
    createGlobalXml2lyInsiderOahGroup ());

  // ... ... ...
}
```

Chapter 39

The OAH atoms collection

These handy general-purpose OAH atoms are used in MusicFormats itself. They are defined in src/oah/oahAtomsCol

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/oah > grep class   oahAtomsCollection.h
class EXP oahValueLessAtomAlias : public oahAtom
class EXP oahMacroAtom : public oahAtom
class EXP oahOptionsUsageAtom : public oahPureHelpValueLessAtom
class EXP oahHelpAtom : public oahPureHelpValueLessAtom
class EXP oahHelpSummaryAtom : public oahPureHelpValueLessAtom
class EXP oahAboutAtom : public oahPureHelpValueLessAtom
class EXP oahVersionAtom : public oahPureHelpValueLessAtom
class EXP oahContactAtom : public oahPureHelpValueLessAtom
class EXP oahBooleanAtom : public oahAtom
class EXP oahTwoBooleansAtom : public oahBooleanAtom
class EXP oahThreeBooleansAtom : public oahBooleanAtom
class EXP oahCombinedBooleansAtom : public oahAtom
class EXP oahCommonPrefixBooleansAtom : public oahAtom
class EXP oahIntegerAtom : public oahAtomStoringAValue
class EXP oahTwoIntegersAtom : public oahIntegerAtom
class EXP oahFloatAtom : public oahAtomStoringAValue
class EXP oahStringAtom : public oahAtomStoringAValue
class EXP oahFactorizedStringAtom : public oahAtom
class EXP oahDefaultedStringAtom : public oahStringAtom
class EXP oahRationalAtom : public oahAtomStoringAValue
class EXP oahNaturalNumbersSetElementAtom : public oahAtomStoringAValue
class EXP oahColorRGBAtom : public oahAtomStoringAValue
class EXP oahIntSetElementAtom : public oahAtomStoringAValue
class EXP oahStringSetElementAtom : public oahAtomStoringAValue
class EXP oahStringToIntMapElementAtom : public oahAtomStoringAValue
class EXP oahStringAndIntegerAtom : public oahAtomStoringAValue
class EXP oahStringAndTwoIntegersAtom : public oahAtomStoringAValue
class EXP oahLengthUnitKindAtom : public oahAtomStoringAValue
class EXP oahLengthAtom : public oahAtomStoringAValue
class EXP oahMidiTempoAtom : public oahAtomStoringAValue
class EXP oahOptionNameHelpAtom : public oahDefaultedStringAtom
class EXP oahQueryOptionNameAtom : public oahPureHelpValueFittedAtom
class EXP oahFindStringAtom : public oahPureHelpValueFittedAtom
```

39.1 OAH macro atoms

A OAH macro atom is a combination, a list of several options under a single name. The `oahMacroAtom` class is defined in src/oah/oahAtomsCollection.h/.cpp:

```
class EXP oahMacroAtom : public oahAtom
{
/*
  a list of atoms
*/

  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    void                      oahValueLessAtomAlias (S_oahAtom atom);

    void                      applyValueLessAtom (std::ostream& os) override;

  private:

    // private fields
    // ------------------------------------------------------

     std::list<S_oahAtom> fMacroValueLessAtomsList;
};
```

Populating field `oahMacroAtom::fMacroValueLessAtomsList` is straightfoward:

```
void oahMacroAtom::oahValueLessAtomAlias (S_oahAtom atom)
{
#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  mfAssert (
    __FILE__, __LINE__,
    atom != nullptr,
    "atom is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  fMacroValueLessAtomsList.push_back (atom);
}
```

Applying the macro atom is done in method `oahMacroAtom::applyElement ()`:

```
void oahMacroAtom::applyValueLessAtom (std::ostream& os)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gEarlyOptions.getTraceEarlyOptions ()) {
    std::stringstream ss;

    ss <<
      "==> option '" << fetchNames () << "' is a oahMacroAtom";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  for (
    std::list<S_oahAtom>::const_iterator i =
      fMacroValueLessAtomsList.begin ();
    i != fMacroValueLessAtomsList.end ();
    ++i
  ) {
    S_oahAtom atom = (*i);

    if (
      // oahAtomStoringAValue?
      S_oahAtomStoringAValue
        atomWithVariable =
          dynamic_cast<oahAtomStoringAValue*>(&(*atom))
    ) {
//        atomWithVariable-> JMI ???
//          applyAtomWithValue (theString, os);
    }
    else {
      // value-less atom
      atom->
        applyValueLessAtom (os);
    }
  } // for
}

  fSelected = true;
```

39.2 A OAH macro atom example

`xml2brl` has the `-auto-utf8, -au8d` option:

```
jacquesmenu@macmini > xml2brl -query auto-utf8d
--- Help for atom "auto-utf8d" in subgroup "Files"
    -auto-utf8d, -au8d
          Combines -auto-output-file-name, -utf8d and -use-encoding-in-file-name
```

This macro options is defined in `src/formatsgeneration/brailleGeneration/brailleGenerationOah.cpp` the following way:

```
void brailleGenerationOahGroup::initializeMacroOptions ()
{
  S_oahSubGroup
    subGroup =
      oahSubGroup::create (
        "Macros",
        "help-braille-generation-macros", "hbgm",
R"()",
      oahElementVisibilityKind::kElementVisibilityWhole,
      this);

  appendSubGroupToGroup (subGroup);

  // create the auto utfd8 macro

  S_oahMacroAtom
    autoUTFd8MacroAtom =
      oahMacroAtom::create (
        "auto-utf8d", "au8d",
        "Combines -auto-output-file-name, -utf8d and -use-encoding-in-file-name");

  subGroup->
    appendAtomToSubGroup (
      autoUTFd8MacroAtom);

  // populate it
  autoUTFd8MacroAtom->
    oahValueLessAtomAlias (
      gGlobalOutputFileOahGroup->getAutoOutputFileNameAtom ());

  fBrailleOutputKindAtom->
    applyAtomWithValue (
      "utf8d",
      gLog);
  autoUTFd8MacroAtom->
    oahValueLessAtomAlias (
      fBrailleOutputKindAtom);

  autoUTFd8MacroAtom->
    oahValueLessAtomAlias (
      fUseEncodingInFileNameAtom);
}
```

39.3 LilyPond octave entry

Pass `lpsr2lilypond` has three options to choose this, all controlling one and the same variable:

```
jacquesmenu@macmini > xml2ly -query absolute
--- Help for atom "absolute" in subgroup "Notes"
    -abs, -absolute
          Use absolute octave entry in the generated LilyPond code.
```

```
jacquesmenu@macmini > xml2ly -query relative
--- Help for atom "relative" in subgroup "Notes"
    -rel, -relative
          Use relative octave entry reference PITCH_AND_OCTAVE in the generated LilyPond
    code.
          PITCH_AND_OCTAVE is made of a diatonic pitch and
          an optional sequence of commas or single quotes.
          It should be placed between double quotes if it contains single quotes, such as:
            -rel "c''".
          The default is to use LilyPond's implicit reference 'f'.
```

```
jacquesmenu@macmini > xml2ly -query fixed
--- Help for atom "fixed" in subgroup "Notes"
    -fixed
          Use fixed octave entry reference PITCH_AND_OCTAVE in the generated LilyPond code
    .
          PITCH_AND_OCTAVE is made of a diatonic pitch and
          an optional sequence of commas or single quotes.
          It should be placed between double quotes if it contains single quotes, such as:
            -fixed "c''"
```

This is done in `src/formatsgeneration/lilypondGeneration/lpsr2lilypondOah.h/.cpp` using a single instance of class `msrOctaveEntryVariable`:

```
class EXP msrOctaveEntryVariable : public smartable
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    std::string           fVariableName;
    msrOctaveEntryKind    fOctaveEntryKind;
};
```

The three classes:

- `lilypondAbsoluteOctaveEntryAtom`

- `lilypondRelativeOctaveEntryAtom`
- `lilypondFixedOctaveEntryAtom`

all contain an alias for an class `msrOctaveEntryVariable` variable:

```
    // private fields
    // ------------------------------------------------------

    msrOctaveEntryVariable&
                          fOctaveEntryKindVariable;
```

The `fOctaveEntryVariable` filed of class `lpsr2lilypondOahGroup` shared be all three options atoms is:

```
    // notes
    // ------------------------------------

    msrOctaveEntryVariable
                          fOctaveEntryVariable;
```

Chapter 40

Measures handling

Measures are presented at section 20.29 [Measures], page 205.

40.1 Voices contents

Class `msrVoice` contains a list of the first elements and a last segment:

```
    std::list<S_msrVoiceElement>
                          fVoiceInitialElementsList;

    // fVoiceLastSegment contains the music
    // not yet stored in fVoiceInitialElementsList,
    // it is thus logically the end of the latter,
    // and is created implicitly for every voice.
    // It is needed 'outside' of the 'std::list<S_msrElement>'
    // because it is not a mere S_msrElement, but a S_msrSegment
    S_msrSegment          fVoiceLastSegment;
```

40.2 Voice elements

The class `msrVoiceElement` sub-classes instances in `fVoiceInitialElementsList` can be of types:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src > grep 'public msrVoiceElement' formats/
   msr/*.h
formats/msr/msrBeatRepeats.h:class EXP msrBeatRepeat : public msrVoiceElement
formats/msr/msrMeasureRepeats.h:class EXP msrMeasureRepeat : public msrVoiceElement
formats/msr/msrRepeats.h:class EXP msrRepeat : public msrVoiceElement
formats/msr/msrMultiMeasureRests.h:class EXP msrMultiMeasureRest : public msrVoiceElement
formats/msr/msrSegments.h:class EXP msrSegment : public msrVoiceElement
```

Class `msrSegment` contains a list of measures:

```
    // the measures in the segment contain the mmusic
    std::list<S_msrMeasure>
                          fSegmentElementsList;
```

class contains a list of mesure elements:

```
    // elements

    std::list<S_msrMeasureElement>
                          fMeasureElementsList;
```

40.3 Measure elements

The class msrMeasureElements sub-classes instances in can be of types:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msr > grep 'public
    msrMeasureElement' *.h
msrBarChecks.h:class EXP msrBarCheck : public msrMeasureElement
msrBarLines.h:class EXP msrBarLine : public msrMeasureElement
msrBarNumberChecks.h:class EXP msrBarNumberCheck : public msrMeasureElement
msrBreaks.h:class EXP msrLineBreak : public msrMeasureElement
msrBreaks.h:class EXP msrPageBreak : public msrMeasureElement
msrClefs.h:class EXP msrClef : public msrMeasureElement
msrCodas.h:class EXP msrCoda : public msrMeasureElement
msrDoubleTremolos.h:class EXP msrDoubleTremolo : public msrMeasureElement
msrEyeGlasses.h:class EXP msrEyeGlasses : public msrMeasureElement
msrFiguredBasses.h:class EXP msrFiguredBass : public msrMeasureElement
msrHarmonies.h:class EXP msrHarmony : public msrMeasureElement
msrHiddenMeasureAndBarLines.h:class EXP msrHiddenMeasureAndBarLine : public
    msrMeasureElement
msrInstruments.h:class EXP msrScordatura : public msrMeasureElement
msrInstruments.h:class EXP msrAccordionRegistration : public msrMeasureElement
msrInstruments.h:class EXP msrHarpPedalsTuning : public msrMeasureElement
msrInstruments.h:class EXP msrPedal : public msrMeasureElement
msrInstruments.h:class EXP msrDamp : public msrMeasureElement
msrInstruments.h:class EXP msrDampAll : public msrMeasureElement
msrKeys.h:class EXP msrKey : public msrMeasureElement
msrLyrics.h:class EXP msrSyllable : public msrMeasureElement
msrMusicXMLSpecifics.h:class EXP msrPrintLayout : public msrMeasureElement
msrRehearsalMarks.h:class EXP msrRehearsalMark : public msrMeasureElement
msrSegnos.h:class EXP msrSegno : public msrMeasureElement
msrDalSegnos.h:class EXP msrDalSegno : public msrMeasureElement
msrStavesDetails.h:class EXP msrStaffDetails : public msrMeasureElement
msrTempos.h:class EXP msrTempo : public msrMeasureElement
msrTimeSignatures.h:class EXP msrTimeSignature : public msrMeasureElement
msrTranspositions.h:class EXP msrOctaveShift : public msrMeasureElement
msrTranspositions.h:class EXP msrTransposition : public msrMeasureElement
msrVoiceStaffChanges.h:class EXP msrVoiceStaffChange : public msrMeasureElement
```

40.4 Appending measure elements to a measure

Appending music elements to a measure is done by method msrMeasure::appendElementToMeasure (), defined in src/formats/msr/msrMeasures.h/.cpp:

```
void msrMeasure::appendElementToMeasure (const S_msrMeasureElement& elem)
{
  int inputLineNumber =
    elem->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceMeasures ()) {
    std::stringstream ss;

    ss <<
      "Appending measure element " <<
      elem->asShortString () <<
      " to measure " <<
      asShortString () <<
      " in voice \"" <<
      fetchMeasureUpLinkToVoice ()->
        getVoiceName () <<
      "\", currentMeasureCurrentAccumulatedWholeNotesDuration: \"" <<
      fCurrentMeasureCurrentAccumulatedWholeNotesDuration.asString () <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // set elem's measure number
  elem->
    setMeasureNumber (
      fBarLineUpLinkToMeasure->getMeasureNumber ());

  // set elem's measure position
  elem->
    setMeasureElementMeasurePosition (
      this,
      fCurrentMeasureCurrentAccumulatedWholeNotesDuration,
      "appendElementToMeasure()");

  fMeasureElementsList.push_back (elem);

  // take elem's sounding whole notes into account JMI ???
if (false) // JMI v0.9.67
  incrementCurrentMeasureCurrentAccumulatedWholeNotesDuration (
    inputLineNumber,
    elem->
      getMeasureElementSoundingWholeNotes ());
}
```

Here is how a harmony instance is appended to a measure:

```
void msrMeasure::appendHarmonyToMeasure (const S_msrHarmony& harmony)
{
  int inputLineNumber =
    harmony->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceHarmonies ()) {
    gLog <<
      "Appending harmony " << harmony->asString () <<
      " to measure " <<
      this->asShortString () <<
      " in segment '" <<
      fMeasureUpLinkToSegment->getSegmentAbsoluteNumber () <<
      "' in voice \"" <<
      fMeasureUpLinkToSegment->
        getSegmentUpLinkToVoice ()->
          getVoiceName () <<
      "\", currentMeasureCurrentAccumulatedWholeNotesDuration: \"" <<
      fCurrentMeasureCurrentAccumulatedWholeNotesDuration.asString () <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // set harmony's measure number
  harmony->
    setMeasureNumber (
      fBarLineUpLinkToMeasure->getMeasureNumber ());

  // append the harmony to the measure elements list
  // DON'T call 'appendElementToMeasure (harmony)':
  // that would override harmony's measure position,
  // which already has the correct value, thus:
  fMeasureElementsList.push_back (harmony);

  // get harmony sounding whole notes
  mfRational
    harmonySoundingWholeNotes =
      harmony->
        getMeasureElementSoundingWholeNotes ();

  // account for harmony duration in measure whole notes
  incrementCurrentMeasureCurrentAccumulatedWholeNotesDuration (
    inputLineNumber,
    harmonySoundingWholeNotes);

  // this measure contains music
  fMeasureContainsMusic = true;
}
```

The task is simpler when appending a harmony to a measure clone, because the clone's harmony's measure number comes from the clone's original:

```
void msrMeasure::appendHarmonyToMeasureClone (const S_msrHarmony& harmony)
{
  int inputLineNumber =
    harmony->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceHarmonies ()) {
    gLog <<
      "Appending harmony " << harmony->asString () <<
      " to measure clone " <<
      this->asShortString () <<
      " in segment clone '" <<
      fMeasureUpLinkToSegment->getSegmentAbsoluteNumber () <<
      "' in voice clone \"" <<
      fMeasureUpLinkToSegment->
        getSegmentUpLinkToVoice ()->
          getVoiceName () <<
      "\", currentMeasureCurrentAccumulatedWholeNotesDuration: \"" <<
      fCurrentMeasureCurrentAccumulatedWholeNotesDuration.asString () <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  ++gIndenter;

  // append the harmony to the measure elements list
  appendElementToMeasure (harmony);

  // fetch harmony sounding whole notes
  mfRational
    harmonySoundingWholeNotes =
      harmony->
        getMeasureElementSoundingWholeNotes ();

  // account for harmony duration in measure whole notes
  incrementCurrentMeasureCurrentAccumulatedWholeNotesDuration (
    inputLineNumber,
    harmonySoundingWholeNotes);

  // this measure contains music
  fMeasureContainsMusic = true;

  --gIndenter;
}
```

40.5 Appending measures to a segment

Measures are appended to a segment by method `msrSegment::appendMeasureToSegment ()` in src/formats/msr/msrSegments.h/.cpp:

```
void msrSegment::appendMeasureToSegment (const S_msrMeasure& measure)
{
  int inputLineNumber =
    measure->getInputStartLineNumber ();

  std::string measureNumber =
    measure->getMeasureNumber ();

  unsigned int segmentElementsListSize =
    fSegmentElementsList.size ();

  std::string currentMeasureNumber =
    segmentElementsListSize == 0
      ? ""
      : fSegmentElementsList.back ()->getMeasureNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceMeasures ()) {
    std::stringstream ss;

    ss <<
      "Appending measure '" << measureNumber <<
      "' to segment " << asString ();

    if (fSegmentElementsList.size () == 0)
      gLog <<
        ", as first measure";
    else
      gLog <<
      ", after measure number '" << currentMeasureNumber << "'";

    gLog <<
      "' in voice \"" <<
      fSegmentUpLinkToVoice->getVoiceName () <<
      "\"" <<
      ", line " << measure->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  if (measureNumber == currentMeasureNumber) {
    std::stringstream ss;

    ss <<
      "appending measure number '" << measureNumber <<
      "' occurs twice in a row in segment " <<
      asString () <<
      " in voice \"" <<
      fSegmentUpLinkToVoice->getVoiceName () <<
      "\"";

//    msrInternalWarning ( // JMI
    msrInternalError (
      gServiceRunData->getInputSourceName (),
      inputLineNumber,
      __FILE__, __LINE__,
      ss.str ());
  }

  // is measure the first one this segment?
  if (segmentElementsListSize == 0) {
    measure->
      setMeasureFirstInSegmentKind (
        msrMeasureFirstInSegmentKind::kMeasureFirstInSegmentKindYes);
  }
  else {
    measure->
      setMeasureFirstInSegmentKind (
        msrMeasureFirstInSegmentKind::kMeasureFirstInSegmentKindNo);
  }

  // is measure the first one it the voice?
  // this is necessary for voice clones,
  // which don't go down the part-staff-voice-segment hierarchy
  if (! fSegmentUpLinkToVoice->getVoiceFirstMeasure ()) {
    // yes, register it as such
    fSegmentUpLinkToVoice->
      setVoiceFirstMeasure (measure);

    measure->
      setMeasureFirstInVoice ();
  }

  // append measure to the segment
  fSegmentElementsList.push_back (measure);
}
```

Calls to method `msrSegment::appendMeasureToSegment ()` occur in:

- method `msrSegment::createAMeasureAndAppendItToSegment ()` called from:

  method `msrVoice::createAMeasureAndAppendItToVoice ()`

- method `msrVoice::appendMeasureCloneToVoiceClone ()` called from:

  visitor method `msr2msrTranslator::visitStart (S_msrMeasure& elt)`

- method `msrMultiMeasureRest::appendMeasureCloneToMultiMeasureRests ()` called from:

  visitor method `msr2msrTranslator::visitStart (S_msrMeasure& elt)`

- method `msrVoice::appendMeasureCloneToVoiceClone ()` called from:

  visitor method `msr2lpsrTranslator::visitStart (S_msrMeasure& elt)`

- method `msrVoice::createNewLastSegmentFromItsFirstMeasureForVoice ()` called from:

  method `msrVoice::handleVoiceLevelRepeatStart ()`,
  method `msrVoice::handleVoiceLevelRepeatEndingStartWithoutExplicitStart ()`,
  method `msrVoice::handleVoiceLevelRepeatEndingStartWithExplicitStart ()`,
  method `msrVoice::createMeasureRepeatFromItsFirstMeasures ()`,
  method `msrVoice::appendPendingMeasureRepeatToVoice ()`,
  method `msrVoice::appendMultiMeasureRestToVoice ()`

40.6 Appending measures to a voice

Method msrVoice::appendMeasureCloneToVoiceClone () does the job in src/formats/msr/msrVoices.h/.cpp.

```
S_msrMeasure msrVoice::createAMeasureAndAppendItToVoice (
  int         inputLineNumber,
  std::string measureNumber,
  msrMeasureImplicitKind
              measureImplicitKind)
{
  fVoiceCurrentMeasureNumber = measureNumber;

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceMeasures ()) {
    std::stringstream ss;

    ss <<
      "Creating measure '" <<
      measureNumber <<
      "' and appending it to voice \"" << getVoiceName () << "\"" <<
      "', line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fCallsCounter++;

  if (
//      true
//        ||
    false
      &&
    (
      fCallsCounter == 2 && getVoiceName ()
        ==
      "Part_POne_HARMONIES_Staff_Voice_Eleven_HARMONIES"
    )
  ) { // POUSSE JMI
    gLog <<
      std::endl <<
      "++++ createAMeasureAndAppendItToVoice() POUSSE, fCallsCounter: " << fCallsCounter
    << " ++++" <<
      std::endl;
    this->print (gLog);
    gLog <<
      std::endl;
  }

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceMeasuresDetails ()) {
    displayVoice (
      inputLineNumber,
      "createAMeasureAndAppendItToVoice() 1");
  }
#endif // MF_TRACE_IS_ENABLED

  ++gIndenter;

  // create the voice last segment if needed
  if (! fVoiceLastSegment) {
    createNewLastSegmentForVoice (
      inputLineNumber,
      "createAMeasureAndAppendItToVoice() 2");
  }

  // append a new measure with given number to voice last segment
  S_msrMeasure
    result =
      fVoiceLastSegment->
        createAMeasureAndAppendItToSegment (
          inputLineNumber,
          measureNumber,
          measureImplicitKind);

  // result is the new voice last appended measure
  fVoiceLastAppendedMeasure = result;

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceMeasuresDetails ()) {
    displayVoice (
      inputLineNumber,
      "createAMeasureAndAppendItToVoice() 3");
  }
#endif // MF_TRACE_IS_ENABLED

  --gIndenter;

  return result;
}
```

40.7 Translating from MXSR to MSR

This is done in src/passes/mxsr2msr/.

Upon the first visit of S_measure, as class msrMeasure instance is created and append to the current part:

```
void mxsr2msrTranslator::visitStart (S_measure& elt)
{
  // ... ... ...

  // take this measure into account
  ++fPartMeasuresCounter;

  // ... ... ...

  // implicit

/*
  Measures with an implicit attribute set to "yes"
  never display a measure number,
  regardless of the measure-numbering setting.
  *
  The implicit attribute is set to "yes" for measures where
  the measure number should never appear, such as pickup
  measures and the last half of mid-measure repeats. The
  value is "no" if not specified.
*/

  std::string
    implicit =
      elt->getAttributeValue ("implicit");

  msrMeasureImplicitKind
    measureImplicitKind =
      msrMeasureImplicitKind::kMeasureImplicitKindNo; // default value

  if       (implicit == "yes") {
    measureImplicitKind =
      msrMeasureImplicitKind::kMeasureImplicitKindYes;
  }
  else  if (implicit == "no") {
    measureImplicitKind =
      msrMeasureImplicitKind::kMeasureImplicitKindNo;
  }
  else {
    if (implicit.size ()) {
      std::stringstream ss;

      ss <<
        "implicit \"" << implicit <<
        "\" is unknown";

      musicxmlError (
        gServiceRunData->getInputSourceName (),
        inputLineNumber,
        __FILE__, __LINE__,
        ss.str ());
    }
  }

  // append a new measure to the current part
  fCurrentPart->
    createAMeasureAndAppendItToPart (
      inputLineNumber,
      fCurrentMeasureNumber,
      measureImplicitKind);

  // ... ... ...
}
```

Upon the second visit of `S_measure`, the last appended measure appended to the current part is finalized:

```
void mxsr2msrTranslator::visitEnd (S_measure& elt)
{
  // ... ... ...

  // finalize current measure in the part,
  // to add skips if necessary and set measure kind
  fCurrentPart->
    finalizeLastAppendedMeasureInPart (
      inputLineNumber);

  // ... ... ...
}
```

40.8 Translating from MXSR to MSR

A new class `msrMeasure` instance is created in `src/passes/mxsr2msr/mxsr2msrTranslator.cpp` upon the first visit of `S_measure`:

```
//______________________________________________________________________________
void mxsr2msrTranslator::visitStart (S_measure& elt)
{
  // ... ... ...

  // append a new measure to the current part
  fCurrentPart->
    createAMeasureAndAppendItToPart (
      inputLineNumber,
      fCurrentMeasureNumber,
      measureImplicitKind);

  // ... ... ...
}
```

This can lead to several class `msrMeasure` instances being created, depending on the MusicXML data. Hence there is no notion of a current measure in this translator.

Method `msrPart::createAMeasureAndAppendItToPart ()` creates and appends a measure to the part harmonies and figured bass staves if relevant, and then cascade s to the part staves:

```
void msrPart::createAMeasureAndAppendItToPart (
  int         inputLineNumber,
  std::string measureNumber,
  msrMeasureImplicitKind
              measureImplicitKind)
{
  // ... ... ...

  // set part current measure number
  fPartCurrentMeasureNumber = measureNumber;

  // create and append measure in all the staves
  for (S_msrStaff staff : fPartAllStavesList) {
    staff->
      createAMeasureAndAppendItToStaff (
        inputLineNumber,
        measureNumber,
        measureImplicitKind);
  } // for

  // ... ... ...
```

40.9 Translating from MSR to MSR

This is done in src/passes/msr2msr/.

40.10 Translating from MSR to LPSR

This is done in src/passes/msr2lpsr/.

40.11 Translating from LPSR to LilyPond

This is done in src/passes/lpsr2lilypond/.

Chapter 41

Finalizations

41.1 Clones vs non-clones finalization

Finalizing clones may be simpler that finalizing a just-created and populated non-clone, due to the information available in the clone's original.

For example, method `msrMeasure::finalizeMeasure ()` delegates part of the job to methods handling the three kinds of voices, respectively:

```
void msrMeasure::finalizeMeasure (
  int                          inputLineNumber,
  msrMeasureRepeatContextKind measureRepeatContextKind,
  std::string         context)
{
  // ... ... ...

    S_msrVoice
      voice =
        fMeasureUpLinkToSegment->
          getSegmentUpLinkToVoice ();

  // ... ... ...

    // delegate to voice kind specific methods
    switch (voice->getVoiceKind ()) {
      case msrVoiceKind::kVoiceKindRegular:
        finalizeRegularMeasure (
          inputLineNumber,
          measureRepeatContextKind,
          context);
        break;

      case msrVoiceKind::kVoiceKindDynamics:
        break;

      case msrVoiceKind::kVoiceKindHarmonies:
        finalizeHarmonyMeasure (
          inputLineNumber,
          measureRepeatContextKind,
          context);
        break;

      case msrVoiceKind::kVoiceKindFiguredBass:
        finalizeFiguredBassMeasure (
          inputLineNumber,
          measureRepeatContextKind,
          context);
        break;
    } // switch

  // ... ... ...
}
```

In the case of harmony and figured bass voices, padding may have to be added to obtain a complete measure. This does not happen for clones of such voices: the padding skips are in the original voice and will be visited and handled without anything special to be done.

41.2 The finalization methods

There is a set of virtual method `finalize* ()` methods in MusicFormats. There basic ones are:

- method `msrPart::finalizePart ()` and method `msrPart::finalizePartClone ()`,
  defined in `src/formats/msr/msrParts.h/.cpp`
- method `msrStaff::finalizeStaff ()`,
  defined in `src/formats/msr/msrStaves.h/.cpp`
- method `msrVoice::finalizeVoice ()`,
  defined in `src/formats/msr/msrVoices.h/.cpp`

- method `msrSegment::finalizeAllTheMeasuresOfSegment ()`,
  defined in `src/formats/msr/msrSegments.h/.cpp`

- method `msrMeasure::finalizeMeasure ()`, method `msrMeasure::finalizeMeasureClone ()` and method `msrMeasure::finalizeRegularMeasure ()`,
  defined in `src/formats/msr/msrMeasures.h/.cpp`

- method `msrChord::finalizeChord ()`,
  defined in `src/formats/msr/msrChords.h/.cpp`

- method `msrTuplets::finalizeTuplet ()`,
  defined in `src/formats/msr/msrTuplets.h/.cpp`
- method `mxsr2msrTranslator::finalizeTupletAndPopItFromTupletsStack ()`,
  defined in `src/passes/mxsr2msr/mxsr2msrTranslator.h.h/.cpp`

- method `msrMeasure::finalizeFiguredBassMeasure ()`,
  defined in `src/formats/msr/msrMeasures.h/.cpp`
- method `msrMeasure::finalizeHarmonyMeasure ()`,
  defined in `src/formats/msr/msrMeasures.h/.cpp`

- method `msr2bsrTranslator::finalizeCurrentMeasureClone ()`,
  defined in `src/passes/msr2bsr/(.h/.cpp`msr2bsrTranslator)

- method `mxsr2msrTranslator::finalizeCurrentChord ()`,
  defined in `src/passes/mxsr2msr/mxsr2msrTranslator.h/.cpp`

Handling repeats is rather complex in MusicFormats. Repeat ends are finalized with these methods:

- method `msrPart::finalizeRepeatEndInPart ()`,
  defined in `src/formats/msr/msrParts.h/.cpp`
- method `msrStaff::finalizeRepeatEndInStaff ()`,
  defined in `src/formats/msr/msrStaves.h/.cpp`
- method `msrVoice::finalizeRepeatEndInVoice ()`,
  defined in `src/formats/msr/msrVoices.h/.cpp`

There are also 'cascading' finalization methods: they propagate finalization going from class `msrPart` towards class `msrVoice`:

- method `msrPart::finalizeLastAppendedMeasureInPart ()`, defined in `src/formats/msr/msrParts.h/.cpp`
- method `msrStaff::finalizeLastAppendedMeasureInStaff ()`, defined in `src/formats/msr/msrStaves.h/.cpp`
- method `msrVoice::finalizeLastAppendedMeasureInVoice ()`, defined in `src/formats/msr/msrVoices.h/.cpp`

- method `msrPart::finalizePartAndAllItsMeasures ()`, defined in `src/formats/msr/msrParts.h/.cpp`
- method `msrVoice::finalizeVoiceAndAllItsMeasures ()`, defined in `src/formats/msr/msrVoices.h/.cpp`

A typical example of cascadingfinalization is method `msrPart::finalizePartAndAllItsMeasures ()`:

```
void msrPart::finalizePartAndAllItsMeasures (
  int inputLineNumber)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceParts ()) {
    std::stringstream ss;

    ss <<
      "Finalizing part clone " <<
      getPartCombinedName () <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceVoices ()) {
    std::stringstream ss;

    ss <<
      "Finalizing all the measures of part \"" <<
      getPartCombinedName () <<
      "\", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  for (
    std::list<S_msrVoice>::const_iterator i = fPartAllVoicesList.begin ();
    i != fPartAllVoicesList.end ();
    ++i
  ) {
    const S_msrVoice& voice = (*i);

    voice->
      finalizeVoiceAndAllItsMeasures (
        inputLineNumber);
  } // for

  // collect the part measures slices from the staves
  collectPartMeasuresSlices (
    inputLineNumber);
}
```

41.3 Finalizing parts

Method `msrPart::finalizePart ()` warns if there are no staves in the part, and otherwise handles them, including cascadingto method `msrStaff::finalizeStaff ()`:

```
void msrPart::finalizePart (
  int inputLineNumber)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceParts ()) {
    std::stringstream ss;

    ss <<
      "Finalizing part " <<
      getPartCombinedName () <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  ++gIndenter;

  if (! getPartStaveNumbersToStavesMap.size ()) {
    std::stringstream ss;

    ss <<
      "Part " <<
      getPartCombinedName () <<
      " appears in the part list, but doesn't contain any staff";

    musicxmlWarning (
      gServiceRunData->getInputSourceName (),
      inputLineNumber,
      ss.str ());
  }

  else {
    // sort the staves to have harmonies above and figured bass below the part
    fPartAllStavesList.sort (
      compareStavesToHaveFiguredBassesBelowCorrespondingPart);

    // finalize the staves
    for (
      std::map<int, S_msrStaff>::const_iterator i =
      getPartStaveNumbersToStavesMap.begin ();
      i != getPartStaveNumbersToStavesMap.end ();
      ++i
    ) {
      S_msrStaff staff = (*i).second;

      staff->
        finalizeStaff (
          inputLineNumber);
    } // for
  }

  // set score instrument names max lengthes if relevant
  setPartInstrumentNamesMaxLengthes ();

  // collect the part measures slices from the staves
  collectPartMeasuresSlices (
    inputLineNumber);

  --gIndenter;
}
```

41.4 Finalizing staves

Method **msrStaff::finalizeStaff ()** cascade s to method **msrVoice::finalizeVoice ()** and collects the staff measures slices:

```
void msrStaff::finalizeStaff (int inputLineNumber)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceStaves ()) {
    std::stringstream ss;

    ss <<
      "Finalizing staff \"" <<
      getStaffName () << "\"" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  ++gIndenter;

  // finalize the voices
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceVoices ()) {
    std::stringstream ss;

    ss <<
      "Finalizing the voices in staff \"" <<
      getStaffName () << "\"" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  for (
    std::map<int, S_msrVoice>::const_iterator i =
      fStaffVoiceNumbersToAllVoicesMap.begin ();
    i != fStaffVoiceNumbersToAllVoicesMap.end ();
    ++i
  ) {
    S_msrVoice
      voice = (*i).second;

    voice->
```

```
      finalizeVoice (
        inputLineNumber);
  } // for

  // collect the staff measures slices from the voices
  collectStaffMeasuresSlices (
    inputLineNumber);

  --gIndenter;
}
```

41.5 Finalizing voices

Method `msrVoice::finalizeVoice ()` handles pending repeats if any and collects the voice measures into a flat list. It does not, however, cascade to finalizing the voice repeats and measures.

```
void msrVoice::finalizeVoice (
  int inputLineNumber)
{
  // ... ... ...

  if (fVoiceHasBeenFinalized) {
    std::stringstream ss;

    ss <<
      "Attempting to finalize  voice \"" <<
      asShortString () <<
      "\" more than once";

    msrInternalError (
      gServiceRunData->getInputSourceName (),
      fInputStartLineNumber,
      __FILE__, __LINE__,
      ss.str ());
  }

  // set part shortest note duration if relevant
  S_msrPart
    voicePart =
      fetchVoiceUpLinkToPart ();

  mfRational
    partShortestNoteWholeNotes =
      voicePart->
        getPartShortestNoteWholeNotes ();

  // ... ... ...

  if (fVoiceShortestNoteWholeNotes < partShortestNoteWholeNotes) {
    // set the voice part shortest note duration
    voicePart->
      setPartShortestNoteWholeNotes (
        fVoiceShortestNoteWholeNotes);

    // set the voice part shortest note tuplet factor // JMI
    voicePart->
      setPartShortestNoteTupletFactor (
        fVoiceShortestNoteTupletFactor);
  }

  // is this voice totally empty? this should be rare...
  if (
    fVoiceInitialElementsList.size () == 0
```

```
      &&
    fVoiceLastSegment->getSegmentElementsList ().size () == 0
  ) {
    std::stringstream ss;

    ss <<
      "Voice \"" <<
      getVoiceName () <<
      "\" is totally empty, no contents ever specified for it" <<
      std::endl;

    musicxmlWarning (
      gServiceRunData->getInputSourceName (),
      inputLineNumber,
      ss.str ());
  }

  // are there pending repeats in the voice repeats stack???
  unsigned int voicePendingRepeatDescrsStackSize =
    fVoicePendingRepeatDescrsStack.size ();

  // ... ... ...

  // collect the voice measures into the flat list
  collectVoiceMeasuresIntoFlatList (
    inputLineNumber);

  fVoiceHasBeenFinalized = true;

  // ... ... ...
}
```

41.6 Finalizing repeats

41.7 Finalizing measures

Method `msrMeasure::finalizeMeasure ()` is not cascaded. It delegates finalization to voice kind specific methods presented in the subsections below, handles pending repeats if any, and assigns positions in the measure to the measure's elements:

```
void msrMeasure::finalizeMeasure (
  int                           inputLineNumber,
  msrMeasureRepeatContextKind measureRepeatContextKind,
  std::string         context)
{
  if (fMeasureHasBeenFinalized) {
    std::stringstream ss;

    ss <<
      "Attempting to finalize measure " <<
      this->asShortString () <<
      " more than once in segment '" <<
      fMeasureUpLinkToSegment->getSegmentAbsoluteNumber () <<
      "', context: " << context <<
      "', measureFinalizationContext: " << fMeasureFinalizationContext <<
      " in voice \"" <<
      fMeasureUpLinkToSegment->
        getSegmentUpLinkToVoice ()->
          getVoiceName () <<
      "\" (" << context << ")" <<
      ", line " << inputLineNumber;

    // ... ... ...

    msrInternalWarning (
      gServiceRunData->getInputSourceName (),
      fInputStartLineNumber,
      ss.str ());
  }

  else {
    S_msrVoice
      voice =
        fMeasureUpLinkToSegment->
          getSegmentUpLinkToVoice ();

    // ... ... ...

    // delegate to voice kind specific methods
    switch (voice->getVoiceKind ()) {
      case msrVoiceKind::kVoiceKindRegular:
        finalizeRegularMeasure (
          inputLineNumber,
          measureRepeatContextKind,
          context);
        break;

      case msrVoiceKind::kVoiceKindDynamics:
        break;

      case msrVoiceKind::kVoiceKindHarmonies:
        finalizeHarmonyMeasure (
          inputLineNumber,
          measureRepeatContextKind,
          context);
        break;
      case msrVoiceKind::kVoiceKindFiguredBass:
        finalizeFiguredBassMeasure (
          inputLineNumber,
          measureRepeatContextKind,
          context);
        break;
    } // switch

    // position in voice
    mfRational
      voicePosition =
        fetchMeasureUpLinkToVoice ()->
          getCurrentVoicePosition ();

    // assign measure' elements measure position
    for (
      std::list<S_msrMeasureElement>::const_iterator i = fMeasureElementsList.begin ();
      i != fMeasureElementsList.end ();
      ++i
    ) {
      S_msrMeasureElement measureElement = (*i);

      measureElement->
        setMeasureElementVoicePosition (
          voicePosition,
          "finalizeMeasure()");
    } // for

    // register finalization
    fMeasureHasBeenFinalized = true;
    fMeasureFinalizationContext = context;
  }
}
```

41.7.1 Finalizing regular measures

```
void msrMeasure::finalizeRegularMeasure (
  int                          inputLineNumber,
  msrMeasureRepeatContextKind measureRepeatContextKind,
  std::string        context)
{
  // fetch the regular voice
  S_msrVoice
    voice =
      fMeasureUpLinkToSegment->
        getSegmentUpLinkToVoice ();

  // fetch the regular voice's part
  S_msrPart
    regularPart =
      voice->
        fetchVoiceUpLinkToPart ();

  mfAssert (
    __FILE__, __LINE__,
    regularPart != nullptr,
    "regularPart is null");

  if (false) { // JMI
    gLog <<
      "---> regularPart: " <<
      std::endl;

    ++gIndenter;
    gLog <<
      regularPart <<
      std::endl;
    --gIndenter;
    gLog << std::endl;
  }

  mfRational
    measureCurrentAccumulatedWholeNotesDurationFromPartMeasuresVector =
      regularPart->
        getPartMeasuresWholeNotessVector () [
           fMeasureOrdinalNumberInVoice - 1 ];

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceMeasures ()) {
    std::stringstream ss;

    ss <<
      "Finalizing regular measure " <<
      this->asShortString () <<
      " in segment '" <<
      fMeasureUpLinkToSegment->getSegmentAbsoluteNumber () <<
      "' in regular voice \"" <<
      voice->getVoiceName () <<
      "\" (" << context << ")" <<
      ", measureCurrentAccumulatedWholeNotesDurationFromPartMeasuresVector: " <<
      measureCurrentAccumulatedWholeNotesDurationFromPartMeasuresVector <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
```

```
#endif // MF_TRACE_IS_ENABLED

  ++gIndenter;

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceMeasuresDetails ()) {
    displayMeasure (
      inputLineNumber,
      "finalizeRegularMeasure() 1");
  }
#endif // MF_TRACE_IS_ENABLED

  padUpToPositionAtTheEndOfTheMeasure (
    inputLineNumber,
    measureCurrentAccumulatedWholeNotesDurationFromPartMeasuresVector);

  // register this measures's length in the part
  S_msrPart
    part =
      this->fetchMeasureUpLinkToPart ();

  part->
    registerOrdinalMeasureNumberWholeNotes (
      inputLineNumber,
      fMeasureOrdinalNumberInVoice,
      fCurrentMeasureCurrentAccumulatedWholeNotesDuration);

  // determine the measure kind
  determineMeasureKindAndPuristNumber (
    inputLineNumber,
    measureRepeatContextKind);

  // pad measure up to whole measure whole notes high tide JMI ???
  switch (fMeasureKind) {
    case msrMeasureKind::kMeasureKindCadenza:
      break;

    case msrMeasureKind::kMeasureKindOverFlowing:
    case msrMeasureKind::kMeasureKindAnacrusis:
    case msrMeasureKind::kMeasureKindRegular:
    case msrMeasureKind::kMeasureKindIncompleteStandalone: // JMI
    case msrMeasureKind::kMeasureKindIncompleteLastMeasure: // JMI
    case msrMeasureKind::kMeasureKindIncompleteLastInRepeatCommonPart: // JMI
    case msrMeasureKind::kMeasureKindIncompleteLastInRepeatHookedEnding: // JMI
    case msrMeasureKind::kMeasureKindIncompleteLastInRepeatHooklessEnding: // JMI
    case msrMeasureKind::kMeasureKindIncompleteNextMeasureAfterCommonPart: // JMI
    case msrMeasureKind::kMeasureKindIncompleteNextMeasureAfterHookedEnding: // JMI
    case msrMeasureKind::kMeasureKindIncompleteNextMeasureAfterHooklessEnding: // JMI
      break;

    case msrMeasureKind::kMeasureKindUnknown:
      // JMI ???
      break;

    case msrMeasureKind::kMeasureKindMusicallyEmpty:
      {
      /* JMI
          */
      }
      break;
  } // switch

  // is there a single note or rest occupying the full measure?
  if (fMeasureLongestNote) {
    if (
      fMeasureLongestNote->getMeasureElementSoundingWholeNotes ()
        ==
      fFullMeasureWholeNotesDuration
    ) {
#ifdef MF_TRACE_IS_ENABLED
      if (gTraceOahGroup->getTraceMeasures ()) {
        gLog <<
          "Note '" <<
          fMeasureLongestNote->asShortString () <<
          "' occupies measure " <<
          this->asShortString () <<
          " fully in segment '" <<
          fMeasureUpLinkToSegment->getSegmentAbsoluteNumber () <<
          "' in voice \"" <<
          voice->getVoiceName () <<
          "\", line " << inputLineNumber <<
          std::endl;

        gWaeHandler->waeTraceWithoutInputLocation (
          __FILE__, __LINE__,
          ss.str ());
      }
#endif // MF_TRACE_IS_ENABLED

      fMeasureLongestNote->
        setNoteOccupiesAFullMeasure ();
    }
  }

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceMeasuresDetails ()) {
    displayMeasure (
      inputLineNumber,
      "finalizeRegularMeasure() 2");
  }
#endif // MF_TRACE_IS_ENABLED

  --gIndenter;
}
```

41.7.2 Finalizing harmonies measures

```
void msrMeasure::finalizeHarmonyMeasure (
  int                             inputLineNumber,
  msrMeasureRepeatContextKind measureRepeatContextKind,
  std::string         context)
{
  // fetch the harmonies voice
  S_msrVoice
    harmoniesVoice =
      fMeasureUpLinkToSegment->
        getSegmentUpLinkToVoice ();

  // fetch the harmonies part
  S_msrPart
    harmoniesPart =
      harmoniesVoice->
        fetchVoiceUpLinkToPart ();

  mfAssert (
    __FILE__, __LINE__,
    harmoniesPart != nullptr,
    "harmoniesPart is null");

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceHarmonies ()) {
    std::stringstream ss;

    ss <<
      "Finalizing harmonies measure " <<
      this->asShortString () <<
      " in segment '" <<
      fMeasureUpLinkToSegment->getSegmentAbsoluteNumber () <<
      "' in harmonies voice \"" <<
      harmoniesVoice->getVoiceName () <<
      "\" (" << context << ")" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  ++gIndenter;

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceHarmoniesDetails ()) {
    displayMeasure (
      inputLineNumber,
      "finalizeHarmonyMeasure() 1");
  }
#endif // MF_TRACE_IS_ENABLED

#ifdef MF_TRACE_IS_ENABLED
  // get the harmoniesPart number of measures
  int
    harmoniesPartNumberOfMeasures =
      harmoniesPart->
        getPartNumberOfMeasures ();

  if (gTraceOahGroup->getTraceHarmonies ()) {
    std::stringstream ss;

    ss <<
      "fMeasureOrdinalNumberInVoice: \"" <<
      fMeasureOrdinalNumberInVoice <<
      ", harmoniesPartNumberOfMeasures: \"" <<
      harmoniesPartNumberOfMeasures;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // the measureCurrentAccumulatedWholeNotesDuration has to be computed
  mfRational
    measureCurrentAccumulatedWholeNotesDurationFromPartMeasuresVector =
      harmoniesPart->
        getPartMeasuresWholeNotessVector () [
           fMeasureOrdinalNumberInVoice - 1 ];

  // handle the harmonies in this measure
  finalizeTheHarmoniesInAHarmoniesMeasure (
    inputLineNumber,
    context);

  // pad the measure up to
   measureCurrentAccumulatedWholeNotesDurationFromPartMeasuresVector
  padUpToPositionAtTheEndOfTheMeasure (
    inputLineNumber,
    measureCurrentAccumulatedWholeNotesDurationFromPartMeasuresVector);

  // determine the measure kind
  determineMeasureKindAndPuristNumber (
    inputLineNumber,
    measureRepeatContextKind);

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceHarmoniesDetails ()) {
    displayMeasure (
      inputLineNumber,
      "finalizeHarmonyMeasure() 2");
  }
#endif // MF_TRACE_IS_ENABLED

  --gIndenter;
}
```

41.7.3 Finalizing figured bass measures

```
void msrMeasure::finalizeFiguredBassMeasure (
  int                           inputLineNumber,
  msrMeasureRepeatContextKind measureRepeatContextKind,
  std::string        context)
{
  // fetch the figured bass voice
  S_msrVoice
    figuredBassVoice =
      fMeasureUpLinkToSegment->
        getSegmentUpLinkToVoice ();

  // fetch the figured bass part
  S_msrPart
    figuredBassPart =
      figuredBassVoice->
        fetchVoiceUpLinkToPart ();

  mfAssert (
    __FILE__, __LINE__,
    figuredBassPart != nullptr,
    "figuredBassPart is null");

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceFiguredBasses ()) {
    std::stringstream ss;

    ss <<
      "Finalizing figured bass measure " <<
      this->asShortString () <<
      " in segment '" <<
      fMeasureUpLinkToSegment->getSegmentAbsoluteNumber () <<
      "' in figured bass voice \"" <<
      figuredBassVoice->getVoiceName () <<
      "\" (" << context << ")" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  ++gIndenter;

#ifdef MF_TRACE_IS_ENABLED
```

```
  if (gTraceOahGroup->getTraceFiguredBassesDetails ()) {
    displayMeasure (
      inputLineNumber,
      "finalizeFiguredBassMeasure() 1");
  }
#endif // MF_TRACE_IS_ENABLED

#ifdef MF_TRACE_IS_ENABLED
  // get the figuredBassPart number of measures
  int
    figuredBassPartNumberOfMeasures =
      figuredBassPart->
        getPartNumberOfMeasures ();

  if (gTraceOahGroup->getTraceHarmonies ()) {
    std::stringstream ss;

    ss <<
      "fMeasureOrdinalNumberInVoice: \"" <<
      fMeasureOrdinalNumberInVoice <<
      ", figuredBassPartNumberOfMeasures: \"" <<
      figuredBassPartNumberOfMeasures;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // the measureCurrentAccumulatedWholeNotesDuration has to be computed
  mfRational
    measureCurrentAccumulatedWholeNotesDuration =
      figuredBassPart->
        getPartMeasuresWholeNotessVector () [
           fMeasureOrdinalNumberInVoice - 1 ];

  // handle the figured bass elements in this measure
  finalizeTheFiguredBassesInAFiguredBassMeasure (
    inputLineNumber,
    context);

  // pad the measure up to fFullMeasureWholeNotesDuration
  padUpToPositionAtTheEndOfTheMeasure (
    inputLineNumber,
    measureCurrentAccumulatedWholeNotesDuration);

  // determine the measure kind
  determineMeasureKindAndPuristNumber (
    inputLineNumber,
    measureRepeatContextKind);

#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceFiguredBassesDetails ()) {
    displayMeasure (
      inputLineNumber,
      "finalizeFiguredBassMeasure() 2");
  }
#endif // MF_TRACE_IS_ENABLED

  --gIndenter;
}
```

41.8 Determining measure positionss

Chapter 42

Tempos handling

Tempos are presented at section 20.17 [Tempos], page 193.

42.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

42.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

42.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

42.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

42.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 43

Notes handling

Notes are presented at section **??** [Notes], page **??**.

43.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

43.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

43.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

43.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

43.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 44

Segments handling

Segments are presented at section 20.36 [Segments], page 211.

The segments concept used by MusicFormats to describe music scores is not apparent to the users of GUI applications, in which music elements are *drawn* on the page. Their need is inherent to the representation of repeats, which contain music elements sequences (the segments) and even other repeats.

ALL SEGMENTS HANDLING in MusicFormats IS DONE INTERNALLY: the class `msrSegment` instances are created in voices and repeats BEHIND THE CURTAINS.

44.1 Segments creation

Instances of class `msrSegment` are created at four places in `src/formats/msr/msrVoices.cpp`:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msr > grep 'msrSegment::create (' 
    *.cpp
msrSegments.cpp:S_msrSegment msrSegment::create (
msrSegments.cpp:      msrSegment::create (
msrSegments.cpp:      msrSegment::create (
msrVoices.cpp:        msrSegment::create (
msrVoices.cpp:    msrSegment::create (
msrVoices.cpp:    msrSegment::create (
msrVoices.cpp:      msrSegment::create (
```

Calls to method `msrSegment::createSegmentNewbornClone ()` occurs only when visiting class `msrSegment` instances in passes:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/passes > grep createSegmentNewbornClone
    */*
msr2bsr/msr2bsrTranslator.cpp:      elt->createSegmentNewbornClone (
msr2lpsr/msr2lpsrTranslator.cpp:    elt->createSegmentNewbornClone (
msr2msr/msr2msrTranslator.cpp:    elt->createSegmentNewbornClone (
```

Method `msrSegment::createSegmentDeepClone ()` is not used at the time of this writing.

Explicit segments creation is thus entirely done in methods inside `src/formats/msr/msrVoices.cpp`: the passes are not aware of this happening.

The first occurrence of method `msrSegment::create ()` is in method `msrVoice::initializeVoice ()`: when a voice is created, a segment is created and stored in its `fVoiceLastSegment` if requested :

```
void msrVoice::initializeVoice (
  msrVoiceCreateInitialLastSegmentKind
    voiceCreateInitialLastSegmentKind)
{
  // ... ... ...

    // create the initial last segment if requested
  switch (voiceCreateInitialLastSegmentKind) {
    case msrVoiceCreateInitialLastSegmentKind::kCreateInitialLastSegmentYes:
      // sanity check // JMI LAST
      mfAssert (
        __FILE__, __LINE__,
        fVoiceLastSegment == nullptr,
        "fVoiceLastSegment is null");

      // create the last segment
      fVoiceLastSegment =
        msrSegment::create (
          fInputStartLineNumber,
          this);

      if (! fVoiceFirstSegment) {
        fVoiceFirstSegment = fVoiceLastSegment;
      }
      break;
    case msrVoiceCreateInitialLastSegmentKind::kCreateInitialLastSegmentNo:
      break;
  } // switch

  // ... ... ...
};
```

Method `msrVoice::createMeasureRepeatFromItsFirstMeasures ()` is presented in section 46 [Measure repeats handling], page 322, and the remaining two are presented in the next sections.

44.1.1 Creating a new last segment for a voice

There is method `msrVoice::createNewLastSegmentForVoice ()`, called at many places in `src/formats/msr/msrVoices.cpp`:

```
void msrVoice::createNewLastSegmentForVoice (
  int    inputLineNumber,
  const std::string& context)
{
  // ... ... ...

  // create the last segment
  fVoiceLastSegment =
    msrSegment::create (
      inputLineNumber,
      this);

  if (! fVoiceFirstSegment) {
    fVoiceFirstSegment = fVoiceLastSegment;
  }

  // ... ... ...
}
```

The calls to method `msrVoice::createNewLastSegmentForVoice ()` are in:

- method msrVoice::createAMeasureAndAppendItToVoice ()

- method msrVoice::appendStaffDetailsToVoice ()

- method msrVoice::addGraceNotesGroupBeforeAheadOfVoiceIfNeeded ()

- method msrVoice::handleVoiceLevelRepeatStart ()
- method msrVoice::handleVoiceLevelRepeatEndWithoutStart ()
- method msrVoice::handleVoiceLevelContainingRepeatEndWithoutStart ()
- method msrVoice::handleVoiceLevelRepeatEndWithStart ()
- method msrVoice::handleVoiceLevelRepeatEndingStartWithoutExplicitStart ()
- method msrVoice::handleVoiceLevelRepeatEndingStartWithExplicitStart ()

- method msrVoice::handleMultiMeasureRestsStartInVoiceClone ()

- method msrVoice::handleHooklessRepeatEndingEndInVoice ()

- method msrVoice::appendBarLineToVoice ()
- method msrVoice::appendSegnoToVoice ()
- method msrVoice::appendCodaToVoice ()
- method msrVoice::appendEyeGlassesToVoice ()
- method msrVoice::appendPedalToVoice ()
- method msrVoice::appendDampToVoice ()
- method msrVoice::appendDampAllToVoice ()

44.1.2 Creating a new last segment for a voice from its first measure

Method msrVoice::createNewLastSegmentFromItsFirstMeasureForVoice () is used at several places in src/formats/msr/msrVoices.cpp:

```
void msrVoice::createNewLastSegmentFromItsFirstMeasureForVoice (
  int           inputLineNumber,
  S_msrMeasure firstMeasure,
  std::string        context)
{
  // create the last segment
  fVoiceLastSegment =
    msrSegment::create (
      inputLineNumber,
      this);

  if (! fVoiceFirstSegment) {
    fVoiceFirstSegment = fVoiceLastSegment;
  }

  // ... ... ...

  // append firstMeasure to fVoiceLastSegment
  fVoiceLastSegment->
    appendMeasureToSegment (firstMeasure);

  // firstMeasure is the new voice last appended measure
  fVoiceLastAppendedMeasure = firstMeasure;

  // is firstMeasure the first one it the voice?
  if (! fVoiceFirstMeasure) {
    // yes, register it as such
    setVoiceFirstMeasure (
      firstMeasure);

    firstMeasure->
      setMeasureFirstInVoice ();
  }

  // ... ... ...
}
```

All the uses of this method concern repeats (section 48 [Repeats handling], page 324), measure repeats (section 46 [Measure repeats handling], page 322) and multi-measure rests(section 47 [Multi-measure rests handling], page 323).

44.2 Appending measures to a segment

Method `msrSegment::assertSegmentElementsListIsNotEmpty ()` is called as a sanity check by many methods in `src/formats/msr/msrSegments.cpp`:

```
void msrSegment::assertSegmentElementsListIsNotEmpty (
  int inputLineNumber) const
{
  if (! fSegmentElementsList.size ()) {
#ifdef MF_TRACE_IS_ENABLED
  if (
    gTraceOahGroup->getTraceMeasuresDetails ()
      ||
    gTraceOahGroup->getTraceSegmentsDetails ()
      ||
    gTraceOahGroup->getTraceRepeatsDetails ()
  ) {
    fSegmentUpLinkToVoice->
      displayVoiceRepeatsStackMultiMeasureRestsMeasureRepeatAndVoice (
        inputLineNumber,
        "assertSegmentElementsListIsNotEmpty()");
  }
#endif // MF_TRACE_IS_ENABLED

    gLog <<
      "assertSegmentElementsListIsNotEmpty()" <<
      ", fSegmentElementsList is empty" <<
      ", segment: " <<
      this->asString () <<
      ", in voice \"" <<
      fSegmentUpLinkToVoice->getVoiceName () <<
      "\"" <<
      "', line " << inputLineNumber <<
      std::endl;

    mfAssert (
      __FILE__, __LINE__,
      false,
      ", fSegmentElementsList is empty");
  }
}
```

One such call is:

```
void msrSegment::appendKeyToSegment (
  const S_msrKey& key)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceKeys ()) {
    std::stringstream ss;

    ss <<
      "Appending key " << key
      " to segment " << asString () <<
    ", in voice \"" <<
    fSegmentUpLinkToVoice->getVoiceName () <<
    "\"";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
  assertSegmentElementsListIsNotEmpty (
    key->getInputStartLineNumber ());
#endif // MF_SANITY_CHECKS_ARE_ENABLED

  ++gIndenter;

  // register key in segments's current measure
  fSegmentElementsList.back ()->
    appendKeyToMeasure (key);

  --gIndenter;
}
```

44.3 Translating from MXSR to MSR

44.4 Translating from MXSR to MSR

This is done in src/passes/mxsr2msr/.

44.5 Translating from MSR to MSR

This is done in src/passes/msr2msr/.

44.6 Translating from MSR to LPSR

This is done in src/passes/msr2lpsr/.

44.7 Translating from LPSR to LilyPond

This is done in `src/passes/lpsr2lilypond/`.

Chapter 45

Beat repeats handling

Beat repeats are presented at section 20.31 [Beat repeats], page 207.

45.1 Translating from MXSR to MSR (src/passes/mxsr2msr/)

45.2 Translating from MXSR to MSR (src/passes/mxsr2msr/)

45.3 Translating from MSR to MSR (src/passes/msr2msr/)

45.4 Translating from MSR to LPSR (src/passes/msr2lpsr/)

45.5 Translating from LPSR to LilyPond (src/passes/lpsr2lilypond/)

Chapter 46

Measure repeats handling

Measure repeats are presented at section 20.32 [Measure repeats], page 208.

46.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

method `msrVoice::createMeasureRepeatFromItsFirstMeasures ()`:

46.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

46.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

46.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

46.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 47

Multi-measure rests handling

Multi-measure rests are presented at section 20.33 [Multi-measure rests], page 209.

Chapter 48

Repeats handling

Repeats are presented at section 20.35 [Repeats], page 210.

48.1 Translating repeats from MXSR to MSR

This is done in src/passes/mxsr2msr/.

The tough part is to handle MusicXML `<barLine/>` markups, since they are meant for `drawing`, and do not structure repeats as such.

Recognizing the structure of repeat relies on the attributes of the barLines. The following enumeration type `s` are defined in src/formats/msr/msrBarLines.h local to class `msrBarLine`:

```
    // location
    enum class msrBarLineLocationKind {
      kBarLineLocationNone,

      kBarLineLocationLeft,
      kBarLineLocationMiddle,
      kBarLineLocationRight // by default
    };
```

```
    // style
    enum class msrBarLineStyleKind {
      kBarLineStyleNone,

      kBarLineStyleRegular,  // by default

      kBarLineStyleDotted, kBarLineStyleDashed, kBarLineStyleHeavy,
      kBarLineStyleLightLight, kBarLineStyleLightHeavy,
      kBarLineStyleHeavyLight, kBarLineStyleHeavyHeavy,
      kBarLineStyleTick, kBarLineStyleShort
    };
```

```
    // repeat direction
    enum class msrBarLineRepeatDirectionKind {
      kBarLineRepeatDirectionNone,
      kBarLineRepeatDirectionForward, kBarLineRepeatDirectionBackward
    };
    // ending type
    enum class msrBarLineEndingTypeKind {
      kBarLineEndingTypeNone,

      msrBarLineEndingTypeKind::kBarLineEndingTypeStart,
      msrBarLineEndingTypeKind::kBarLineEndingTypeStop,
      msrBarLineEndingTypeKind::kBarLineEndingTypeDiscontinue
    };
```

```
    // category
    enum class msrBarLineCategoryKind {
      kBarLineCategory_UNKNOWN_,

      kBarLineCategoryStandalone,

      kBarLineCategoryRepeatStart, kBarLineCategoryRepeatEnd,

      kBarLineCategoryHookedEndingStart, kBarLineCategoryHookedEndingEnd,
      kBarLineCategoryHooklessEndingStart, kBarLineCategoryHooklessEndingEnd
    };
```

```
    // segno
    enum class msrBarLineHasSegnoKind {
      kBarLineHasSegnoYes, kBarLineHasSegnoNo
    };
```

```
    // coda
    enum class msrBarLineHasCodaKind {
      kBarLineHasCodaYes, kBarLineHasCodaNo
    };
```

```
    // repeat winged
    enum class msrBarLineRepeatWingedKind {
      kBarLineRepeatWingedNone,

      kBarLineRepeatWingedStraight, kBarLineRepeatWingedCurved,
      kBarLineRepeatWingedDoubleStraight, kBarLineRepeatWingedDoubleCurved
    };
```

The attributes of `<barline/>` are deciphered upon the first visit of `S_barline` in `src/passes/mxsr2msr/mxsr2msrTranslator.cpp`:

```
void mxsr2msrTranslator::visitStart ( S_barline& elt )
{
  // ... ... ...

  // location

  {
    std::string
      location =
        elt->getAttributeValue ("location");

    fCurrentBarLineLocationKind =
      msrBarLineLocationKind::kBarLineLocationRight; // by default

    if       (location == "left") {
      fCurrentBarLineLocationKind = msrBarLineLocationKind::kBarLineLocationLeft;
    }
    else  if (location == "middle") {
      fCurrentBarLineLocationKind = msrBarLineLocationKind::kBarLineLocationMiddle;
    }
    else if  (location == "right") {
      fCurrentBarLineLocationKind = msrBarLineLocationKind::kBarLineLocationRight;
    }
    else {
      std::stringstream ss;

      ss <<
        "barLine location \"" << location <<
        "\" is unknown, using 'right' by default";

 // JMI   musicxmlError (
      musicxmlWarning (
        gServiceRunData->getInputSourceName (),
        inputLineNumber,
 //       __FILE__, __LINE__,
        ss.str ());
    }
  }

  fOnGoingBarLine = true;
}
```

Then the class `msrBarLine` instance is created upon the second visit of `S_barline`:

```
void mxsr2msrTranslator::visitEnd ( S_barline& elt )
{
  // ... ... ...

  // create the barLine
  S_msrBarLine
    barLine =
      msrBarLine::create (
        inputLineNumber,
        fCurrentBarLineLocationKind,
        fCurrentBarLineStyleKind,
        fCurrentBarLineRepeatDirectionKind,
        fCurrentBarLineEndingTypeKind,
        fCurrentBarLineEndingNumber,
        fCurrentBarLineTimes,
        msrBarLineCategoryKind::kBarLineCategory_UNKNOWN_, // will be set afterwards
        fCurrentBarLineHasSegnoKind,
        fCurrentBarLineHasCodaKind,
        fCurrentBarLineRepeatWingedKind);

  // ... ... ...

  // wait until its category is defined
  // to append the barLine to the current segment

  // handle the barLine according to: JMI
  // http://www.musicxml.com/tutorial/the-midi-compatible-part/repeats/

  Bool barLineHasBeenHandled = false;

  switch (fCurrentBarLineLocationKind) {
    case msrBarLineLocationKind::kBarLineLocationNone:
      // should not occur
      break;

    case msrBarLineLocationKind::kBarLineLocationLeft:
      if (
        fCurrentBarLineEndingTypeKind
          ==
        msrBarLineEndingTypeKind::kBarLineEndingTypeStart
      ) {
        // ending start, don't know yet whether it's hooked or hookless
        // ------------------------------------------------------
        if (! fCurrentBarLineEndingNumber.size ()) {
          musicxmlWarning (
            gServiceRunData->getInputSourceName (),
            inputLineNumber,
            "mandatory ending number is missing, assuming \"1\"");

          fCurrentBarLineEndingNumber = "1";
        }

        // don't know yet whether repeat ending start barLine is hooked or hookless
        // remember it in fCurrentRepeatEndingStartBarLine,
        fCurrentRepeatEndingStartBarLine = barLine;

        // handle the repeat ending start
        handleRepeatEndingStart (barLine);

        barLineHasBeenHandled = true;
      }

      else if (
        fCurrentBarLineRepeatDirectionKind
          ==
        msrBarLineRepeatDirectionKind::kBarLineRepeatDirectionForward
      ) {
        // repeat start
        // ------------------------------------------------------
        // set the barLine category
        barLine->
          setBarLineCategory (
            msrBarLineCategoryKind::kBarLineCategoryRepeatStart);

        // handle the repeat start
        handleRepeatStart (barLine);

        barLineHasBeenHandled = true;
      }
      break;

    case msrBarLineLocationKind::kBarLineLocationMiddle:
      // JMI ???
      break;

    case msrBarLineLocationKind::kBarLineLocationRight:
      {
        if (
          fCurrentBarLineEndingTypeKind == msrBarLineEndingTypeKind::
kBarLineEndingTypeStop
            &&
          fCurrentBarLineEndingNumber.size () != 0
        ) {
          // hooked ending end
          // ------------------------------------------------------
          // set current barLine ending start category
          fCurrentRepeatEndingStartBarLine->
            setBarLineCategory (
              msrBarLineCategoryKind::kBarLineCategoryHookedEndingStart);

          // set this barLine's category
          barLine->
            setBarLineCategory (
              msrBarLineCategoryKind::kBarLineCategoryHookedEndingEnd);

          // handle the repeat hooked ending end
          handleRepeatHookedEndingEnd (barLine);

          barLineHasBeenHandled = true;
        }

        else if (
          fCurrentBarLineRepeatDirectionKind
            ==
          msrBarLineRepeatDirectionKind::kBarLineRepeatDirectionBackward
        ) {
          // repeat end
          // ------------------------------------------------------

          // set this barLine's category
          barLine->
            setBarLineCategory (
              msrBarLineCategoryKind::kBarLineCategoryRepeatEnd);

          // handle the repeat end
          handleRepeatEnd (barLine);

          barLineHasBeenHandled = true;
        }

        else if (
          fCurrentBarLineEndingTypeKind == msrBarLineEndingTypeKind::
  kBarLineEndingTypeDiscontinue
            &&
          fCurrentBarLineEndingNumber.size () != 0
        ) {
          // hookless ending end
          // ------------------------------------------------------
          // set current barLine ending start category
          fCurrentRepeatEndingStartBarLine->
            setBarLineCategory (
              msrBarLineCategoryKind::kBarLineCategoryHooklessEndingStart);

          // set this barLine's category
          barLine->
            setBarLineCategory (
              msrBarLineCategoryKind::kBarLineCategoryHooklessEndingEnd);

          // handle the repeat hookless ending end
          handleRepeatHooklessEndingEnd (barLine);

          barLineHasBeenHandled = true;
        }

        // forget about current repeat ending start barLine
        fCurrentRepeatEndingStartBarLine = nullptr;
      }
      break;
  } // switch

  // set the barLine category to stand alone if not yet handled
  if (! barLineHasBeenHandled) {
    switch (fCurrentBarLineStyleKind) {
      case msrBarLineStyleKind::kBarLineStyleRegular:
      case msrBarLineStyleKind::kBarLineStyleDotted:
      case msrBarLineStyleKind::kBarLineStyleDashed:
      case msrBarLineStyleKind::kBarLineStyleHeavy:
      case msrBarLineStyleKind::kBarLineStyleLightLight:
      case msrBarLineStyleKind::kBarLineStyleLightHeavy:
      case msrBarLineStyleKind::kBarLineStyleHeavyLight:
      case msrBarLineStyleKind::kBarLineStyleHeavyHeavy:
      case msrBarLineStyleKind::kBarLineStyleTick:
      case msrBarLineStyleKind::kBarLineStyleShort:
        barLine->
          setBarLineCategory (
            msrBarLineCategoryKind::kBarLineCategoryStandalone);

        // append the bar line to the current part
        // ... ... ...

        fCurrentPart->
          appendBarLineToPart (barLine);

        barLineHasBeenHandled = true;
        break;

      case msrBarLineStyleKind::kBarLineStyleNone:
        std::stringstream ss;

        ss <<
          "barLine " <<
          barLine->asString () <<
          " has no barLine style";

        musicxmlWarning (
          gServiceRunData->getInputSourceName (),
          inputLineNumber,
    //      __FILE__, __LINE__,
          ss.str ());
        break;
    } // switch
  }

  // has this barLine been handled?
  if (! barLineHasBeenHandled) {
    std::stringstream ss;

    ss << std::left <<
      "cannot handle a barLine containing: " <<
      barLine->asString ();

    msrInternalWarning (
      gServiceRunData->getInputSourceName (),
      inputLineNumber,
      ss.str ());
  }

  fOnGoingBarLine = false;
}
```

48.2 Translating repeats from MXSR to MSR

This is done in src/passes/mxsr2msr/.

48.3 Translating repeats from MSR to MSR

This is done in src/passes/msr2msr/.

48.4 Translating repeats from MSR to LPSR

This is done in src/passes/msr2lpsr/.

48.5 Translating repeats from LPSR to LilyPond

This is done in `src/passes/lpsr2lilypond/`.

Chapter 49

Voices handling

Voices are presented at section 20.28 [Voices], page 204.

49.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

49.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

49.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

49.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

49.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 50

Staves handling

Staves are presented at section 20.26 [Staves], page 203.

50.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

50.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

50.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

50.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

50.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 51

Parts handling

Parts are presented at section 20.25 [Parts], page 202.

51.1 Parts browsing

Method `msrPart::browseData ()` defined in `src/formats/msr/msrParts.h/.cpp` is pecular in that it imposes a *partial order* on the part staves browsing:

```
void msrPart::browseData (basevisitor* v)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "% ==> msrPart::browseData ()";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

#ifdef MF_TRACE_IS_ENABLED // JMI
  if (gMsrOahGroup->getTraceMsrVisitors ()) { // JMI TEMP
    gLog <<
      "++++++++ fPartAllStavesList.size(): " <<
      fPartAllStavesList.size () <<
      std::endl;

    if (fPartAllStavesList.size ()) {
      for (S_msrStaff staff : fPartAllStavesList) {
        gLog <<
          std::endl <<
          "+++++++++ staff: ++++++++" <<
          " \"" << staff->getStaffName () << "\"" <<
          std::endl;
      } // for
    }

    gLog <<
      "++++++++ fPartNonHarmoniesNorFiguredBassStavesList.size(): " <<
      fPartNonHarmoniesNorFiguredBassStavesList.size () <<
      std::endl;

    if (fPartNonHarmoniesNorFiguredBassStavesList.size ()) {
      for (S_msrStaff staff : fPartNonHarmoniesNorFiguredBassStavesList) {
        gLog <<
          std::endl <<
          "+++++++++ staff: ++++++++" <<
          " \"" << staff->getStaffName () << "\"" <<
          std::endl;
      } // for
    }
  }
#endif // MF_TRACE_IS_ENABLED

  /* don't enforce any order here, leave it to the client thru sorting JMI */

  // browse the part harmonies staff if any right now, JMI
  // to place it before the corresponding part
  if (fPartHarmoniesStaff) {
    msrBrowser<msrStaff> browser (v);
    browser.browse (*fPartHarmoniesStaff);
  }

  // browse all non harmonies and non figured bass staves
  for (S_msrStaff staff : fPartNonHarmoniesNorFiguredBassStavesList) {
    // browse the staff
    msrBrowser<msrStaff> browser (v);
    browser.browse (*staff);
  } // for

  // browse the part figured bass staff if any only now, JMI
  // to place it after the corresponding part
  if (fPartFiguredBassStaff) {
    msrBrowser<msrStaff> browser (v);
    browser.browse (*fPartFiguredBassStaff);
  }

//   // browse all the part staves JMI
//   for (S_msrStaff staff : fPartAllStavesList) {
//     if (staff != fPartHarmoniesStaff && staff != fPartFiguredBassStaff) {
//       // browse the staff
//       msrBrowser<msrStaff> browser (v);
//       browser.browse (*staff);
//     }
//   } // for
}
```

51.2 Translating from MXSR to MSR (src/passes/mxsr2msr/)

51.3 Translating from MXSR to MSR (src/passes/mxsr2msr/)

51.4 Translating from MSR to MSR (src/passes/msr2msr/)

51.5 Translating from MSR to LPSR (src/passes/msr2lpsr/)

51.6 Translating from LPSR to LilyPond (src/passes/lpsr2lilypond/)

Chapter 52

Part groups handling

MusicFormats part groups are presented at chapter 20.24 [Part groups], page 202.

In MusicXML, part groups can overlap, even though no one seems ever to have needed that. That seems to be more a feature in the Finale handling of MusicXMLexport that a true musical need.

MSR does not support overlapping part group. Handling part groups is done in src/passes/mxsr2msr/mxsr2msrTran
where overlapping groups are identified and rejected:

```
jacquesmenu@macmini > xml2ly partgroups/OverlappingPartGroups.xml
### MusicXML ERROR ### partgroups/OverlappingPartGroups.xml:169:
There are overlapping part groups, namely:
  '2' -=> PartGroup_6 ('2', partGroupName "1
2"), lines 164..169
and
  '1' -=> PartGroup_2 ('1', partGroupName ""), lines 76..170

Please contact the maintainers of MusicFormats (see option '-c, -contact'):
  either you found a bug in the xml2ly converter,
  or this MusicXML data is the first-ever real-world case
  of a score exhibiting overlapping part groups
  std::exception caught: mfException:
There are overlapping part groups, namely:
  '2' -=> PartGroup_6 ('2', partGroupName "1
2"), lines 164..169
and
  '1' -=> PartGroup_2 ('1', partGroupName ""), lines 76..170

Please contact the maintainers of MusicFormats (see option '-c, -contact'):
  either you found a bug in the xml2ly converter,
  or this MusicXML data is the first-ever real-world case
  of a score exhibiting overlapping part groups

  Error message(s) were issued for input line 169
  ### xml2ly gIndenter final value: 1 ###
### Conversion from MusicXML to LilyPond failed ###
```

class mxsrPartGroup contains:

```
struct mxsrPartGroup : public smartable
{
/*
  positions represent the order in which the parts appear in <part-list />
*/

  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    int                      fStartInputLineNumber;
    int                      fStopInputLineNumber;

    int                      fPartGroupNumber; // may be reused later

    S_msrPartGroup           fPartGroup;

    int                      fStartPosition;
    int                      fStopPosition;
};
```

Part groups numbers number re-used and they can be nested, so there is an implicit outer-most part group at the top of their hierachy, attached to the class `msrScore`:

```
class EXP mxsr2msrSkeletonBuilder :
  // ... ... ...

    // an implicit outer-most part group has to be created to contain everything,
    // since there can be parts out of any explicit part group in MusicXML
    S_mxsrPartGroup      fImplicitOuterMxsrPartGroup;
    S_msrPartGroup            fImplicitOuterPartGroup;

    void                      createImplicitOuterPartGroup ();

    // part groups numbers can be re-used, they're no identifier
    // we use a map to access them by part group number
    int                       fPartGroupsCounter;
    std::vector<S_mxsrPartGroup>
                              fPartGroupDescsVector;
    std::map<int, S_mxsrPartGroup>
                              fAllMxsrPartGroupsMap;
    std::map<int, S_mxsrPartGroup>
                              fStartedMxsrPartGroupsMap;

  // ... ... ...
```

Chapter 53

Scores handling

Scores are presented at section 20.23 [Scores], page 202.

53.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

53.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

53.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

53.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

53.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 54

Books handling

Books are presented at section 20.22 [Books], page 201.

54.1 Translating from MXSR to MSR (src/passes/mxsr2msr/)

54.2 Translating from MXSR to MSR (src/passes/mxsr2msr/)

54.3 Translating from MSR to MSR (src/passes/msr2msr/)

54.4 Translating from MSR to LPSR (src/passes/msr2lpsr/)

54.5 Translating from LPSR to LilyPond (src/passes/lpsr2lilypond/)

Chapter 55

Ornaments handling

Ornaments are presented at section **??** [Ornaments], page **??**.

55.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

55.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

55.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

55.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

55.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 56

Ties handling

Ties are presented at section **??** [Ties], page **??**.

56.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

56.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

56.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

56.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

56.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 57

Dynamics handling

Dynamics are presented at section **??** [Dynamics], page **??**.

57.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

57.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

57.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

57.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

57.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 58

Beams handling

Beams are presented at section ?? [Beams], page ??.

58.1 Translating from MXSR to MSR (src/passes/mxsr2msr/)

58.2 Translating from MXSR to MSR (src/passes/mxsr2msr/)

58.3 Translating from MSR to MSR (src/passes/msr2msr/)

58.4 Translating from MSR to LPSR (src/passes/msr2lpsr/)

58.5 Translating from LPSR to LilyPond (src/passes/lpsr2lilypond/)

Chapter 59

Slurs handling

Slurs are presented at section **??** [Slurs], page **??**.

59.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

59.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

59.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

59.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

59.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 60

Grace notes groups handling

Grace notes groups are presented at section 20.38 [Grace notes groups], page 213.

60.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

60.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

60.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

60.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

60.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 61

Chords handling

Chords are presented at section 20.39 [Chords], page 213.

61.1 Translating from MXSR to MSR (src/passes/mxsr2msr/)

61.2 Translating from MXSR to MSR (src/passes/mxsr2msr/)

61.3 Translating from MSR to MSR (src/passes/msr2msr/)

61.4 Translating from MSR to LPSR (src/passes/msr2lpsr/)

61.5 Translating from LPSR to LilyPond (src/passes/lpsr2lilypond/)

Chapter 62

Tuplets handling

Tuplets are presented at section 20.40 [Tuplets], page 213.

62.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

62.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

62.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

62.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

62.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 63

Harmonies handling

Harmonies are presented at section 20.42 [Harmonies], page 215.

The useful options here are:

- option `-trace-harmonies, -tharms`
- option `-display-msr1-skeleton, -dmsrskel`
- option `-display-msr1, -dmsr1`
- option `-display-msr1-full, -dmsr1full`
- option `-display-msr2, -dmsr2`
- option `-display-msr2-full, -dmsr2ull`
- option `-display-lpsr, -dlpsr`
- option `-display-lpsr-full, -dlpsrull`

63.1 Harmonies in MusicXML

In the MusicXML view of music scores, harmonies are simply *drawn* at the current music position, so to say.

The `<harmony/>` markup occurs at the measure level in MusicXML, as notes do, but they apply at the part level, even though this is not explictly stated, since there is no staff nor voice information associated with them:

```
<!--
  The harmony elements are based on Humdrum's **harm
  encoding, extended to support chord symbols in popular
  music as well as functional harmony analysis in classical
  music.

  If there are alternate harmonies possible, this can be
  specified using multiple harmony elements differentiated
  by type. Explicit harmonies have all note present in the
  music; implied have some notes missing but implied;
  alternate represents alternate analyses.

  The harmony object may be used for analysis or for
  chord symbols. The print-object attribute controls
  whether or not anything is printed due to the harmony
  element. The print-frame attribute controls printing
  of a frame or fretboard diagram. The print-style entity
  sets the default for the harmony, but individual elements
  can override this with their own print-style values.

  A harmony element can contain many stacked chords (e.g.
  V of II). A sequence of harmony-chord entities is used
  for this type of secondary function, where V of II would
  be represented by a harmony-chord with a V function
  followed by a harmony-chord with a II function.
-->
<!ENTITY % harmony-chord "((root | function), kind,
  inversion?, bass?, degree*)">

<!ELEMENT harmony ((%harmony-chord;)+, frame?,
  offset?, %editorial;, staff?)>
<!ATTLIST harmony
    type (explicit | implied | alternate) #IMPLIED
    %print-object;
    print-frame  %yes-no; #IMPLIED
    %print-style;
    %placement;
    %optional-unique-id;
>
```

The harmony `<function/>`, defined in MusicXML, is a string. It is stored in the class `msrHarmony`, but nothing more. No example of that is present in the MusicXML sample files this author could find, actually.

```
The <function> element represents classical functional harmony with an indication like I,
    II, III rather than C, D, E. It represents the Roman numeral part of a functional
    harmony rather than the complete function itself. It has been deprecated as of
    MusicXML 4.0 in favor of the <numeral> element.
```

The MusicXML 4.0 `<numeral/>` markup is described at https://www.w3.org/2021/06/musicxml40/musicxml-refe
elements/numeral/. Examples are:

```
<harmony default-y="-80">
   <numeral>
      <numeral-root text="IV">4</numeral-root>
   </numeral>
   <inversion>1</inversion>
</harmony>

<harmony default-y="20">
   <numeral>
      <numeral-root text="III">3</numeral-root>
      <numeral-alter location="left">-1</numeral-alter>
   </numeral>
   <kind halign="center" text="">major</kind>
</harmony>
```

63.2 Harmonies in MSR

Harmonies need special treatment since we need to determine their position in a harmony `msrVoice`, taking `offsets` if any into account.

Harmonies are handled this way, using denormalization:

- harmonies are attached to `msrNote` instances,
- they are placed in specific `msrVoice` instances, whose field `fVoiceKind::contains` `msrVoiceKind::kVoiceKindH`
- there are also attached to `msrNote`, `msrChord` and `msrTuplet` (denormalization);

In class `msrNote`, there is:

```
    // harmonies
    void                      appendHarmonyToNote (
                                const S_msrHarmony& harmony);

    const std::list<S_msrHarmony>&
                              getNoteHarmoniesList () const
                                  { return fNoteHarmoniesList; }

    // ... ... ...

    // harmonies
    // ------------------------------------------------------

    std::list<S_msrHarmony>
                              fNoteHarmoniesList;

    // ... ... ...
```

63.3 Harmonies staves and voices

Every class `msrVoice` instance in MusicFormats belongs to an class `msrStaff` instance. Staves are created specifically to hold harmonies voices, using specific numbers defined in src/formats/msr/msrParts.h:

Since a voice belongs to a `msrStaff` in MSR, there are `msrStave` instance to contain them,

```
  public:

    // constants
    // ------------------------------------------------------

    static const int K_PART_HARMONIES_STAFF_NUMBER;
    static const int K_PART_HARMONIES_VOICE_NUMBER;
```

In class `msrStaff`, there is:

```
    void                      registerHarmoniesVoiceByItsNumber (
                                int           inputLineNumber,
                                const S_msrVoice& voice);
```

Class `msrPart` also contains:

```
   // harmonies

    S_msrVoice                createPartHarmoniesVoice (
                                int                     inputLineNumber,
                                const std::string& currentMeasureNumber);

    void                      appendHarmonyToPart (
                                const S_msrVoice&   harmonySupplierVoice,
                                const S_msrHarmony& harmony);

    void                      appendHarmonyToPartClone (
                                const S_msrVoice&    harmonySupplierVoice,
                                const S_msrHarmony& harmony);
```

```
    // harmonies

    S_msrStaff                fPartHarmoniesStaff;
    S_msrVoice                fPartHarmoniesVoice;
```

63.4 Harmonies staves creation

This is done in src/passes/mxsr2msr/mxsr2msrSkeletonBuilder.cpp.h/.cpp:

```
S_msrVoice mxsr2msrSkeletonBuilder::createPartHarmoniesVoiceIfNotYetDone (
  int         inputLineNumber,
  const S_msrPart&  part)
{
  // is the harmonies voice already present in part?
  S_msrVoice
    partHarmoniesVoice =
      part->
        getPartHarmoniesVoice ();

  if (! partHarmoniesVoice) {
    // create the harmonies voice and append it to the part
    partHarmoniesVoice =
      part->
        createPartHarmoniesVoice (
          inputLineNumber,
          fCurrentMeasureNumber);
  }

  return partHarmoniesVoice;
}
```

Method msrPartcreatePartHarmoniesVoice creates the part harmonies staff and the part harmonies voice, and then registers the latter in the former:

```
S_msrVoice msrPart::createPartHarmoniesVoice (
  int     inputLineNumber,
  std::string currentMeasureNumber)
{
  // ... ... ...

  // create the part harmonies staff
  int partHarmoniesStaffNumber =
    msrPart::K_PART_HARMONIES_STAFF_NUMBER;

  // ... ... ...

  fPartHarmoniesStaff =
    addHarmoniesStaffToPart (
      inputLineNumber);

  // ... ... ...

  // create the part harmonies voice
  int partHarmoniesVoiceNumber =
    msrPart::K_PART_HARMONIES_VOICE_NUMBER;

  // ... ... ...

  fPartHarmoniesVoice =
    msrVoice::create (
      inputLineNumber,
      msrVoiceKind::kVoiceKindHarmonies,
      partHarmoniesVoiceNumber,
      msrVoiceCreateInitialLastSegmentKind::kCreateInitialLastSegmentYes,
      fPartHarmoniesStaff);

  // register the part harmonies voice in part harmonies staff
  fPartHarmoniesStaff->
    registerVoiceInStaff (
      inputLineNumber,
      fPartHarmoniesVoice);

  // ... ... ...

  return fPartHarmoniesVoice;
}
```

63.5 Translating harmonies from MXSR to MSR

This is done in src/passes/mxsr2msr/.

The MSR score skeleton created in src/passes/mxsr2msr/mxsr2msrSkeletonBuilder.h/.cpp contains the part groups, parts, staves and voices, as well as the number of measures. The voices do not contain any music elements yet.

A harmony belongs to a <part/> in MusicXML, but we sometimes need to have it attached to a note. When visiting an S_harmony element, field mxsr2msrSkeletonBuilder::fThereAreHarmoniesToBeAttachedToCurrentNot
it used to account for that:

```
void mxsr2msrSkeletonBuilder::visitStart ( S_harmony& elt )
{
#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting S_harmony" <<
      ", harmoniesVoicesCounter = " << fHarmoniesVoicesCounter <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  /* JMI ???
    several harmonies can be attached to a given note,
    leading to as many harmonies voices in the current part
  */

  // take harmonies voice into account
  ++fHarmoniesVoicesCounter; // UNUSED JMI

  fThereAreHarmoniesToBeAttachedToCurrentNote = true;
}
```

Upon the second visit of class msrNote, the part harmonies voice is created if harmonies are not to be ignored due to option option -ignore-musicxml-harmonies, -oharms and it has not been created yet:

```
void mxsr2msrSkeletonBuilder::visitEnd ( S_note& elt )
{
  // ... ... ...

  // are there harmonies attached to the current note?
  if (fThereAreHarmoniesToBeAttachedToCurrentNote) {
    if (gGlobalMxsr2msrOahGroup->getIgnoreHarmonies ()) {
#ifdef MF_TRACE_IS_ENABLED
      if (gTraceOahGroup->getTraceHarmonies ()) {
        gLog <<
          "Ignoring the harmonies" <<
          ", line " <<
          inputLineNumber <<
          std::endl;

        gWaeHandler->waeTraceWithoutInputLocation (
          __FILE__, __LINE__,
          ss.str ());
      }
#endif // MF_TRACE_IS_ENABLED
    }
    else {
      // create the part harmonies voice if not yet done
      S_msrVoice
        partHarmoniesVoice =
          createPartHarmoniesVoiceIfNotYetDone (
            inputLineNumber,
            fCurrentPart);
    }

    fThereAreHarmoniesToBeAttachedToCurrentNote = false;
  }

  // ... ... ...
}
```

Creating the part harmonies voice is delegated to the part:

```
S_msrVoice mxsr2msrSkeletonBuilder::createPartHarmoniesVoiceIfNotYetDone (
  int         inputLineNumber,
  const S_msrPart&  part)
{
  // is the harmonies voice already present in part?
  S_msrVoice
    partHarmoniesVoice =
      part->
        getPartHarmoniesVoice ();

  if (! partHarmoniesVoice) {
    // create the harmonies voice and append it to the part
    partHarmoniesVoice =
      part->
        createPartHarmoniesVoice (
          inputLineNumber,
          fCurrentMeasureNumber);
  }

  return partHarmoniesVoice;
}
```

63.5.1 First S_harmony visit

The first visit of **S_harmony** initializes the fields storing values to be gathered visiting subelements:

```
void mxsr2msrTranslator::visitStart ( S_harmony& elt )
{
  int inputLineNumber =
    elt->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting S_harmony" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  ++fHarmoniesVoicesCounter;

  fCurrentHarmonyInputLineNumber       = inputLineNumber;

  fCurrentHarmonyRootDiatonicPitchKind = msrDiatonicPitchKind::kDiatonicPitch_UNKNOWN_;
  fCurrentHarmonyRootAlterationKind    = msrAlterationKind::kAlterationNatural;
  fCurrentHarmonyKind                  = msrHarmonyKind::kHarmony_UNKNOWN_;
  fCurrentHarmonyKindText              = "";
  fCurrentHarmonyInversion             = K_HARMONY_INVERSION_NONE;
  fCurrentHarmonyBassDiatonicPitchKind = msrDiatonicPitchKind::kDiatonicPitch_UNKNOWN_;
  fCurrentHarmonyBassAlterationKind    = msrAlterationKind::kAlterationNatural;
  fCurrentHarmonyDegreeValue           = -1;
  fCurrentHarmonyDegreeAlterationKind  = msrAlterationKind::kAlterationNatural;

  fCurrentHarmonyWholeNotesOffset = mfRational (0, 1);

  fOnGoingHarmony = true;
}
```

63.5.2 Second `S_harmony` visit

Upon the second visit of `S_harmony` , a class `msrHarmony` instance is created, populated and appended to `mxsr2msrTranslatorfPendingHarmoniesList`.
The voice uplink will be set later, hence the use of method `msrHarmony::create ()`:

```
void mxsr2msrTranslator::visitEnd ( S_harmony& elt )
{
  // ... ... ...

  if (gGlobalMxsr2msrOahGroup->getIgnoreHarmonies ()) {
 #ifdef MF_TRACE_IS_ENABLED
    if (gTraceOahGroup->getTraceHarmonies ()) {
      gLog <<
        "Ignoring harmony" <<
        ", line " <<
        inputLineNumber;

      gWaeHandler->waeTraceWithoutInputLocation (
        __FILE__, __LINE__,
        ss.str ());
    }
#endif // MF_TRACE_IS_ENABLED
  }
  else {
   // create the harmony
#ifdef MF_TRACE_IS_ENABLED
    if (gTraceOahGroup->getTraceHarmoniesDetails ()) {
      gLog <<
        "Creating a harmony" <<
        ", line " << inputLineNumber << ":" <<
        std::endl;

      // ... ... ...
    }
#endif // MF_TRACE_IS_ENABLED

    S_msrHarmony
      harmony =
        msrHarmony::create (
          fCurrentHarmonyInputLineNumber,
          // no harmoniesUpLinkToVoice yet

          fCurrentHarmonyRootQuarterTonesPitchKind,

          fCurrentHarmonyKind,
          fCurrentHarmonyKindText,

          fCurrentHarmonyInversion,

          fCurrentHarmonyBassQuarterTonesPitchKind,

          mfRational (1, 1),            // harmonySoundingWholeNotes,
                                      // will be set upon next note handling
          mfRational (1, 1),            // harmonyDisplayWholeNotes,
                                      // will be set upon next note handling
          fCurrentHarmoniesStaffNumber,
          msrTupletFactor (1, 1),     // will be set upon next note handling
          fCurrentHarmonyWholeNotesOffset);

    // append pending harmony degrees if any to the harmony
    if (! fCurrentHarmonyDegreesList.size ()) {
#ifdef MF_TRACE_IS_ENABLED
      if (gTraceOahGroup->getTraceHarmoniesDetails ()) {
        musicxmlWarning (
          gServiceRunData->getInputSourceName (),
          inputLineNumber,
          "harmony has no degrees contents");
      }
#endif // MF_TRACE_IS_ENABLED
    }

    else {
      // handle harmony degrees if any
      while (fCurrentHarmonyDegreesList.size ()) {
        S_msrHarmonyDegree
          harmonyDegree =
            fCurrentHarmonyDegreesList.front ();

        // ... ... ...

        // append it to harmony's degrees list
        harmony->
          appendHarmonyDegreeToHarmony (
            harmonyDegree);

        // remove it from the list
        fCurrentHarmonyDegreesList.pop_front ();
      } // while
    }

    // attach the current frame if any to the harmony
    if (fCurrentFrame) {
      harmony->setHarmonyFrame (fCurrentFrame);
    }

    // append the harmony to the pending harmonies list
    fPendingHarmoniesList.push_back (harmony);
  }

  fOnGoingHarmony = false;
}
```

63.5.3 Attaching msrHarmony instances to notes

msrHarmony

The contents of mxsr2msrTranslatorfPendingHarmoniesList is attached to the class msrNote instance in method mxsr2msrTranslator::populateNote ():

```
void mxsr2msrTranslator::populateNote (
  int        inputLineNumber,
  const S_msrNote& newNote)
{
  // ... ... ...

  // handle the pending harmonies if any
  if (fPendingHarmoniesList.size ()) {
    // get voice to insert harmonies into
    S_msrVoice
      voiceToInsertHarmoniesInto =
        fCurrentPart->
          getPartHarmoniesVoice ();

    // ... ... ...

    handlePendingHarmonies (
      newNote,
      voiceToInsertHarmoniesInto);

    // reset harmony counter
    fHarmoniesVoicesCounter = 0;
  }
}
```

63.5.4 Populating msrHarmony instances

msrHarmony

The class msrHarmony instances are populated further in src/formats/msr/mxsr2msrTranslator.cpp and attached to the note by method msrNote::appendHarmonyToNote ():

```
void mxsr2msrTranslator::handlePendingHarmonies (
  const S_msrNote&  newNote,
  const S_msrVoice& voiceToInsertInto)
{
  // ... ... ...

  mfRational
    newNoteSoundingWholeNotes =
      fCurrentNote->
        getMeasureElementSoundingWholeNotes (),
    newNoteDisplayWholeNotes =
      fCurrentNote->
        getNoteDisplayWholeNotes ();

  while (fPendingHarmoniesList.size ()) { // recompute at each iteration
    S_msrHarmony
      harmony =
        fPendingHarmoniesList.front ();

    /*
      MusicXML harmonies don't have a duration,
      and MSR could follow this line, but LilyPond needs one...
      So:
        - we register all harmonies with the duration of the next note
        - they will be sorted by position in the measure in finalizeMeasure(),
            at which time their duration may be shortened
            so that the offsets values are enforced
            and they don't overflow the measure
        It is VITAL that harmonies measures be finalized
        AFTER the corresponding measure in the regular voice,
        since the current sounding whole notes of the latter is needed for that
    */

    // set the harmony's sounding whole notes
    harmony->
      setMeasureElementSoundingWholeNotes (
        newNoteSoundingWholeNotes,
        "mxsr2msrTranslator::handlePendingHarmonies()");

    // set the harmony's display whole notes JMI useless???
    harmony->
      setHarmonyDisplayWholeNotes (
        newNoteDisplayWholeNotes);

    // set the harmony's tuplet factor
    harmony->
      setHarmonyTupletFactor (
        msrTupletFactor (
          fCurrentNoteActualNotes,
          fCurrentNoteNormalNotes));

    // attach the harmony to newNote
    fCurrentNote->
      appendHarmonyToNote (
        harmony);

    // get the part harmonies voice
    S_msrVoice
      partHarmoniesVoice =
        fCurrentPart->
          getPartHarmoniesVoice ();

#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
    mfAssert (
      __FILE__, __LINE__,
      partHarmoniesVoice != nullptr,
      "partHarmoniesVoice is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

    // set the harmony's voice upLink
    // only now that we know which harmonies voice will contain it
    harmony->
      setHarmoniesUpLinkToVoice (
        partHarmoniesVoice);

/* JMI v0.9.67
    // append the harmony to the part harmonies voice
    partHarmoniesVoice->
      appendHarmonyToVoice (
        harmony);
*/
    // don't append the harmony to the part harmonies voice // BLARK
    // before the note itself has been appended to the voice

    // remove the harmony from the list
    fPendingHarmoniesList.pop_front ();
  } // while
}
```

63.5.5 First S_harmony visit

msrHarmony

Method `msrNote::appendHarmonyToNote ()` is where the harmony's note uplink is set:

```
void msrNote::appendHarmonyToNote (const S_msrHarmony& harmony)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceHarmonies ()) {
    gLog <<
      "Appending harmony " <<
      harmony->asString () <<
      " to the harmonies list of " <<
      asString () <<
      ", line " << fInputStartLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // update the harmony whole notes if it belongs to a tuplet ??? utf8.xml JMI

  fNoteHarmoniesList.push_back (harmony);

  // register this note as the harmony note upLink
  harmony->
    setHarmonyUpLinkToNote (this);
}
```

When a harmony is attached to a note that is a chord member, we have to attach it to the chord too, to facilitate setting its measure position when setting the chord's one.

```
void mxsr2msrTranslator::copyNoteHarmoniesToChord (
  const S_msrNote&  note,
  const S_msrChord& chord)
{
  // copy note's harmony if any from the first note to chord

  const std::list<S_msrHarmony>&
    noteHarmoniesList =
      note->getNoteHarmoniesList ();

  if (noteHarmoniesList.size ()) {
    std::list<S_msrHarmony>::const_iterator i;
    for (i = noteHarmoniesList.begin (); i != noteHarmoniesList.end (); ++i) {
      S_msrHarmony harmony = (*i);

#ifdef MF_TRACE_IS_ENABLED
      if (gTraceOahGroup->getTraceHarmonies ()) {
        gLog <<
          "Copying harmony '" <<
          harmony->asString () <<
          "' from note " << note->asString () <<
          " to chord '" << chord <<
          '\'' <<
          std::endl;

        gWaeHandler->waeTraceWithoutInputLocation (
          __FILE__, __LINE__,
          ss.str ());
      }
#endif // MF_TRACE_IS_ENABLED

```

```
      chord->
        appendHarmonyToChord (harmony);

    } // for
  }
}
```

63.5.6 Inserting msrHarmony instances in the part harmonies voice

msrHarmony

Inserting the harmonies in the part harmonies voice is done in method msrVoice::appendNoteToVoice () in src/formats/msr/msrNotes.cpp:

```
void msrVoice::appendNoteToVoice (const S_msrNote& note)
{
  // ... ... ...

  // are there harmonies attached to this note? // BLARK
  const std::list<S_msrHarmony>&
    noteHarmoniesList =
      note->
        getNoteHarmoniesList ();

  if (noteHarmoniesList.size ()) {
    // get the current part's harmonies voice
    S_msrVoice
      partHarmoniesVoice =
        part->
          getPartHarmoniesVoice ();

    for (S_msrHarmony harmony : noteHarmoniesList) {
      // append the harmony to the part harmonies voice
      partHarmoniesVoice->
        appendHarmonyToVoice (
          harmony);
    } // for
  }

  // ... ... ...
};
```

63.6 Translating harmonies from MSR to MSR

This is done in src/passes/msr2msr/.

In src/passes/msr2msr/msr2msrTranslator.cpp, a newborn clone of the harmony is created upon the first visit, stored in msr2msrTranslatorfCurrentHarmonyClone, and appended to the current non grace note clone, the current chord clone or to the current voice clone, if the latter is a harmonies voice:

```
void msr2msrTranslator::visitStart (S_msrHarmony& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrHarmony '" <<
      elt->asString () <<
      ", fOnGoingNonGraceNote: " << fOnGoingNonGraceNote <<
      ", fOnGoingChord: " << fOnGoingChord <<
      ", fOnGoingHarmoniesVoice: " << fOnGoingHarmoniesVoice <<
      ", fOnGoingHarmony: " << fOnGoingHarmony <<
      "', line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create a harmony newborn clone
  fCurrentHarmonyClone =
    elt->
      createHarmonyNewbornClone (
        fCurrentVoiceClone);

  if (fOnGoingNonGraceNote) {
    // register the harmony in the current non-grace note clone
    fCurrentNonGraceNoteClone->
      appendHarmonyToNote (
        fCurrentHarmonyClone);

    // don't append the harmony to the part harmony,
    // this has been done in pass2b // JMI ???
  }

  else if (fOnGoingChord) {
    // register the harmony in the current chord clone
    fCurrentChordClone->
      appendHarmonyToChord (fCurrentHarmonyClone); // JMI
  }

  else if (fOnGoingHarmoniesVoice) {
  /* JMI
    // get the harmony whole notes offset
    mfRational
      harmonyWholeNotesOffset =
        elt->getHarmonyWholeNotesOffset ();

    // is harmonyWholeNotesOffset not equal to 0?
    if (harmonyWholeNotesOffset.getNumerator () != 0) {
      // create skip with duration harmonyWholeNotesOffset
      S_msrNote
        skip =
          msrNote::createSkipNote (
            elt->                getInputStartLineNumber (),
            "666", // JMI elt->                getHarmoniesMeasureNumber (),
            elt->                getHarmonyDisplayWholeNotes (), // would be 0/1 otherwise
     JMI
            elt->                getHarmonyDisplayWholeNotes (),
            0, // JMI elt->                getHarmonyDotsNumber (),
            fCurrentVoiceClone-> getRegularVoiceStaffSequentialNumber (), // JMI
            fCurrentVoiceClone-> getVoiceNumber ());

      // append it to the current voice clone
      // to 'push' the harmony aside
      fCurrentVoiceClone->
        appendNoteToVoice (skip);
    }
*/

    // append the harmony to the current voice clone
    fCurrentVoiceClone->
      appendHarmonyToVoiceClone (
        fCurrentHarmonyClone);
  }

  else {
    std::stringstream ss;

    ss <<
      "harmony is out of context, cannot be handled: " <<
      elt->asShortString ();

    msrInternalError (
      gServiceRunData->getInputSourceName (),
      elt->getInputStartLineNumber (),
      __FILE__, __LINE__,
      ss.str ());
  }

  fOnGoingHarmony = true;
}
```

There are only fields updates upon the second visit:

```
void msr2msrTranslator::visitEnd (S_msrHarmony& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> End visiting msrHarmony '" <<
      elt->asString () <<
      '\'' <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fCurrentHarmonyClone = nullptr;
  fOnGoingHarmony = false;
}
```

63.7 Translating harmonies from MSR to LPSR

This is done in src/passes/msr2lpsr/.

The same occurs in src/passes/msr2lpsr/msr2lpsrTranslator.cpp: a newborn clone of the harmony is created and appended to the current non grace note clone, the current chord clone or to the current voice clone, if the latter is a harmonies voice: :

```
void msr2lpsrTranslator::visitStart (S_msrHarmony& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrHarmony '" <<
      elt->asString () <<
      ", onGoingNonGraceNote: " << fOnGoingNonGraceNote <<
      ", onGoingChord: " << fOnGoingChord <<
      ", onGoingHarmoniesVoice: " << fOnGoingHarmoniesVoice <<
      ", onGoingHarmony: " << fOnGoingHarmony <<
      "', line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create a harmony newborn clone
  fCurrentHarmonyClone =
    elt->
      createHarmonyNewbornClone (
        fCurrentVoiceClone);

  if (fOnGoingNonGraceNote) {
    // register the harmony in the current non-grace note clone
    fCurrentNonGraceNoteClone->
      appendHarmonyToNote (
        fCurrentHarmonyClone);

    // don't append the harmony to the part harmony,
    // this has been done in pass2b // JMI ???
  }

  else if (fOnGoingChord) {
    // register the harmony in the current chord clone
    fCurrentChordClone->
      appendHarmonyToChord (fCurrentHarmonyClone); // JMI
  }

  else if (fOnGoingHarmoniesVoice) {
  /* JMI
    // get the harmony whole notes offset
    mfRational
      harmonyWholeNotesOffset =
        elt->getHarmonyWholeNotesOffset ();

    // is harmonyWholeNotesOffset not equal to 0?
    if (harmonyWholeNotesOffset.getNumerator () != 0) {
      // create skip with duration harmonyWholeNotesOffset
      S_msrNote
        skip =
          msrNote::createSkipNote (
            elt->                  getInputStartLineNumber (),
            "666", // JMI elt->                  getHarmoniesMeasureNumber (),
            elt->                  getHarmonyDisplayWholeNotes (), // would be 0/1 otherwise
     JMI
            elt->                  getHarmonyDisplayWholeNotes (),
            0, // JMI elt->                  getHarmonyDotsNumber (),
            fCurrentVoiceClone-> getRegularVoiceStaffSequentialNumber (), // JMI
            fCurrentVoiceClone-> getVoiceNumber ());

      // append it to the current voice clone
      // to 'push' the harmony aside
      fCurrentVoiceClone->
        appendNoteToVoice (skip);
    }
*/

    // append the harmony to the current voice clone
    fCurrentVoiceClone->
      appendHarmonyToVoiceClone (
        fCurrentHarmonyClone);
  }

  else {
    std::stringstream ss;

    ss <<
      "harmony is out of context, cannot be handled: " <<
      elt->asShortString ();

    msrInternalError (
      gServiceRunData->getInputSourceName (),
      elt->getInputStartLineNumber (),
      __FILE__, __LINE__,
      ss.str ());
  }

  fOnGoingHarmony = true;
}
```

Here too, there are only fields updates upon the second visit of `S_msrHarmony` instances:

```
void msr2lpsrTranslator::visitEnd (S_msrHarmony& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> End visiting msrHarmony '" <<
      elt->asString () <<
      '\'' <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fCurrentHarmonyClone = nullptr;
  fOnGoingHarmony = false;
}
```

63.8 Translating harmonies from LPSR to LilyPond

This is done in src/passes/lpsr2lilypond/.

There is only one visit of class `msrHarmony` instances in src/passes/lpsr2lilypond/lpsr2lilypondTranslator.cpp
The LilyPond code is generated only if the harmony belongs to a voice: this is where denormalization ends in the workflow:

```
void lpsr2lilypondTranslator::visitStart (S_msrHarmony& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  {
    Bool
      traceMsrVisitors =
        gMsrOahGroup->
          getTraceMsrVisitors (),
      generateMsrVisitingInformation =
        gGlobalLpsr2lilypondOahGroup->
          getGenerateMsrVisitingInformation ();

    if (traceMsrVisitors || generateMsrVisitingInformation) {
      std::stringstream ss;

      ss <<
        "% --> Start visiting msrHarmony '" <<
        elt->asString () <<
        '\'' <<
        ", fOnGoingNotesStack.size (): \"" <<
        fOnGoingNotesStack.size () <<
        ", fOnGoingChord: \"" <<
        fOnGoingChord <<
        ", fOnGoingHarmoniesVoice: \"" <<
        fOnGoingHarmoniesVoice <<
        ", line " << elt->getInputStartLineNumber () <<
      std::endl;

      if (traceMsrVisitors) {
        gLog << ss.str ();
      }

      if (generateMsrVisitingInformation) {
        fLilypondCodeStream << ss.str ();
      }
    }
  }
#endif // MF_TRACE_IS_ENABLED

  if (fOnGoingNotesStack.size () > 0) {
  /* JMI
#ifdef MF_TRACE_IS_ENABLED
    if (gTraceOahGroup->getTraceHarmonies ()) {
      fLilypondCodeStream <<
        "%{ fOnGoingNotesStack.size () S_msrHarmony JMI " <<
        elt->asString () <<
        " %}";

      gWaeHandler->waeTraceWithoutInputLocation (
        __FILE__, __LINE__,
        ss.str ());
    }
#endif // MF_TRACE_IS_ENABLED
*/
  }

  else if (fOnGoingChord) { // JMI
  }

  else if (fOnGoingHarmoniesVoice) {
    // actual LilyPond code generation
    fLilypondCodeStream <<
      harmonyAsLilypondString (elt) <<
      ' ';

    // generate the input line number as comment if relevant
    if (
      gGlobalLpsr2lilypondOahGroup->getInputStartLineNumbers ()
        ||
      gGlobalLpsr2lilypondOahGroup->getGenerateMeasurePositions ()
    ) {
      generateInputLineNumberAndOrMeasurePositionAsAComment (
        elt);
    }
  }
}
```

Chapter 64

Figured bass handling

Figured bass elements are presented at section ?? [Figured bass elements], page ??.

The useful options here are:

- option `-trace-figured-bass, -tfigbass`
- option `-display-msr1-skeleton, -dmsrskel`
- option `-display-msr1, -dmsr1`
- option `-display-msr1-full, -dmsr1full`
- option `-display-msr2, -dmsr2`
- option `-display-msr2-full, -dmsr2full`
- option `-display-lpsr, -dlpsr`
- option `-display-lpsr-full, -dlpsrfull`

64.1 Figured bass in MusicXML

In the MusicXML view of figured bass, figured bass elements are simply *drawn* at the current music position, so to say.

```
<!--
  Figured bass elements take their position from the first
  regular note (not a grace note or chord note) that follows
  in score order. The optional duration element is used to
  indicate changes of figures under a note.

  Figures are ordered from top to bottom. A figure-number is
  a number. Values for prefix and suffix include plus and
  the accidental values sharp, flat, natural, double-sharp,
  flat-flat, and sharp-sharp. Suffixes include both symbols
  that come after the figure number and those that overstrike
  the figure number. The suffix values slash, back-slash, and
  vertical are used for slashed numbers indicating chromatic
  alteration. The orientation and display of the slash usually
  depends on the figure number. The prefix and suffix elements
  may contain additional values for symbols specific to
  particular figured bass styles. The value of parentheses
  is "no" if not present.
-->
<!ELEMENT figured-bass (figure+, duration?, %editorial;)>
```

```
<!ATTLIST figured-bass
    %print-style;
    %printout;
    parentheses %yes-no; #IMPLIED
    %optional-unique-id;
>
<!ELEMENT figure
  (prefix?, figure-number?, suffix?, extend?, %editorial;)>
<!ELEMENT prefix (#PCDATA)>
<!ATTLIST prefix
    %print-style;
>
<!ELEMENT figure-number (#PCDATA)>
<!ATTLIST figure-number
    %print-style;
>
<!ELEMENT suffix (#PCDATA)>
<!ATTLIST suffix
    %print-style;
>
```

64.2 Figured bass description

Figured bass is represented in MSR by classes defined in src/formats/msr/msrFiguredBasses.h/.cpp. There is class msrFiguredBass:

```
class EXP msrFiguredBass : public msrMeasureElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrNote             fFiguredBassUpLinkToNote;
    S_msrVoice            fFiguredBassUpLinkToVoice; // for use in figured bass voices JMI

    mfRational              fFiguredBassDisplayWholeNotes;

    msrFiguredBassParenthesesKind
                          fFiguredBassParenthesesKind;

    std::list<S_msrBassFigure> fFiguredBassFiguresList;

    msrTupletFactor       fFiguredBassTupletFactor;
};
```

The figured bass figures are defined in:

```
class EXP msrBassFigure : public msrElement
{
  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // upLinks
    S_msrPart             fFigureUpLinkToPart;

    msrBassFigurePrefixKind
                          fFigurePrefixKind;
    int                   fFigureNumber;
    msrBassFigureSuffixKind
                          fFigureSuffixKind;
};
```

Figured bass elements need special treatment since we need to determine their position in a figured bass voice. This is different than MusicXML, where they are simply *drawn* at the current music position, so to say.

They are handled this way:

- figured bass elements are stored in class `msrNote`:
- they are also stored in class `msrPart` and class `msrChord` and class `msrTuplet` (denormalization);

In class `msrNote`, there is:

```
    // figured bass
    void                  appendFiguredBassToNote (
                            const S_msrFiguredBass& figuredBass);

    const std::list<S_msrFiguredBass>&
                          getNoteFiguredBassesList () const
                              { return fNoteFiguredBassesList; }

    // ... ... ...

    // figured bass
    // ------------------------------------------------------

    std::list<S_msrFiguredBass>
                          fNoteFiguredBassesList;
```

64.3 Figured bass staves and voices

Every class `msrVoice` instance in MusicFormats belongs to an class `msrStaff` instance. Staves are created specifically to hold figured bass voices, using specific numbers defined in `src/formats/msr/msrParts.h`:

```
  public:

    // constants
    // ------------------------------------------------------

    // ... ... ...

    #define msrPart::K_PART_FIGURED_BASS_STAFF_NUMBER  20
    #define msrPart::K_PART_FIGURED_BASS_VOICE_NUMBER  21
```

In class `msrStaff`, there is:

```
     void                  registerFiguredBassVoiceByItsNumber (
                            int          inputLineNumber,
                            const S_msrVoice& voice);
```

Class `msrPart` also contains:

```
    // figured bass

    S_msrVoice              createPartFiguredBassVoice (
                              int     inputLineNumber,
                              std::string currentMeasureNumber);

    void                    appendFiguredBassToPart (
                              const S_msrVoice&             figuredBassSupplierVoice,
                             S_msrFiguredBass figuredBass);

    void                    appendFiguredBassToPartClone (
                              const S_msrVoice&             figuredBassSupplierVoice,
                              const S_msrFiguredBass& figuredBass);
```

```
    // figured bass

    S_msrStaff              fPartFiguredBassStaff;
    S_msrVoice              fPartFiguredBassVoice;
```

64.4 Figured bass staves creation

This is done in src/passes/mxsr2msr/mxsr2msrSkeletonBuilder.cpp.h/.cpp:

```
S_msrVoice mxsr2msrSkeletonBuilder::createPartFiguredBassVoiceIfNotYetDone (
  int         inputLineNumber,
  const S_msrPart&  part)
{
  // is the figured bass voice already present in part?
  S_msrVoice
    partFiguredBassVoice =
      part->
        getPartFiguredBassVoice ();

  if (! partFiguredBassVoice) {
    // create the figured bass voice and append it to the part
    partFiguredBassVoice =
      part->
        createPartFiguredBassVoice (
          inputLineNumber,
          fCurrentMeasureNumber);
  }

  return partFiguredBassVoice;
}
```

Method msrPart::createPartFiguredBassVoice () creates the part figured bass staff and the part figured bass voice, and then registers the latter in the former:

```
S_msrVoice msrPart::createPartFiguredBassVoice (
  int     inputLineNumber,
  std::string currentMeasureNumber)
{
  // ... ... ...

  // create the part figured bass staff
  int partFiguredBassStaffNumber =
    msrPart::K_PART_FIGURED_BASS_STAFF_NUMBER;

  // ... ... ...

  fPartFiguredBassStaff =
    addHFiguredBassStaffToPart (
      inputLineNumber);

  // ... ... ...

  // create the figured bass voice
  int partFiguredBassVoiceNumber =
    msrPart::K_PART_FIGURED_BASS_VOICE_NUMBER;

  // ... ... ...

  fPartFiguredBassVoice =
    msrVoice::create (
      inputLineNumber,
      msrVoiceKind::kVoiceKindFiguredBass,
      partFiguredBassVoiceNumber,
      msrVoiceCreateInitialLastSegmentKind::kCreateInitialLastSegmentYes,
      fPartFiguredBassStaff);

  // register the figured bass voice in the part figured bass staff
  fPartFiguredBassStaff->
    registerVoiceInStaff (
      inputLineNumber,
      fPartFiguredBassVoice);

  // ... ... ...

  return fPartFiguredBassVoice;
}
```

64.5 Translating figured bass from MXSR to MSR

This is done in src/passes/mxsr2msr/, and this is where the class msrFiguredBass instances are created. There several methods for Figured bass elements creation:

```
jacquesmenu@macmini:~/musicformats-git-dev/src/representations/msr > grep create
    msrFiguredBasses.h
    static SMARTP<msrBassFigure> create (
    SMARTP<msrBassFigure> createFigureNewbornClone (
    SMARTP<msrBassFigure> createFigureDeepClone ( // JMI ???
    static SMARTP<msrFiguredBass> create (
    static SMARTP<msrFiguredBass> create (
    SMARTP<msrFiguredBass> createFiguredBassNewbornClone (
    SMARTP<msrFiguredBass> createFiguredBassDeepClone ();
```

The MSR score skeleton created in src/passes/mxsr2msr/mxsr2msrSkeletonBuilder.h/.cpp contains the part groups, parts, staves and voices, as well as the number of measures. The voices do not contain any music elements yet.

A figured bass element belongs to `<part/>` in MusicXML, but we sometimes need to have it attached to a note.
Fieldmxsr2msrSkeletonBuilderfThereAreFiguredBassToBeAttachedToCurrentNote it used when visiting an S_FiguredBass element to account for that:

```
void mxsr2msrSkeletonBuilder::visitStart ( S_figured_bass& elt )
{
#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting S_figured_bass" <<
      ", figuredBassVoicesCounter = " << fFiguredBassVoicesCounter <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  /* JMI
    several figured bass elements can be attached to a given note,
    leading to as many figured bass voices in the current part JMI TRUE???
  */

  // take figured bass voice into account
  ++fFiguredBassVoicesCounter;

  fThereAreFiguredBassToBeAttachedToCurrentNote = true;
}
```

Upon the second visit of class `msrNote`, the part figured bass voice is created if figured bass elements are not to be ignored due to option option `-ignore-musicxml-figured-bass, -ofigbass` and it has not been created yet:

```
void mxsr2msrSkeletonBuilder::visitEnd ( S_note& elt )
{
  // ... ... ...

  // are there figured bass attached to the current note?
  if (fThereAreFiguredBassToBeAttachedToCurrentNote) {
    if (gGlobalMxsr2msrOahGroup->getIgnoreFiguredBasses ()) {
#ifdef MF_TRACE_IS_ENABLED
      if (gTraceOahGroup->getTraceFiguredBasses ()) {
        gLog <<
          "Ignoring the figured bass elements" <<
          ", line " <<
          inputLineNumber <<
          std::endl;

        gWaeHandler->waeTraceWithoutInputLocation (
          __FILE__, __LINE__,
          ss.str ());
      }
#endif // MF_TRACE_IS_ENABLED
    }
    else {
      // create the part figured bass voice if not yet done
      S_msrVoice
        partFiguredBassVoice =
          createPartFiguredBassVoiceIfNotYetDone (
            inputLineNumber,
            fCurrentPart);
    }

    fThereAreFiguredBassToBeAttachedToCurrentNote = false;

  // ... ... ...
}
```

Creating the part figured bass voice is delegated to the part:

```
S_msrVoice mxsr2msrSkeletonBuilder::createPartFiguredBassVoiceIfNotYetDone (
  int          inputLineNumber,
  const S_msrPart&  part)
{
  // is the figured bass voice already present in part?
  S_msrVoice
    partFiguredBassVoice =
      part->
        getPartFiguredBassVoice ();

  if (! partFiguredBassVoice) {
    // create the figured bass voice and append it to the part
    partFiguredBassVoice =
      part->
        createPartFiguredBassVoice (
          inputLineNumber,
          fCurrentMeasureNumber);
  }

  return partFiguredBassVoice;
}
```

64.5.1 First S_figured_bass visit

The first visit of `S_figured_bass` initializes the fields storing values to be gathered visiting subelements:

```
void mxsr2msrTranslator::visitStart ( S_figured_bass& elt )
{
  int inputLineNumber =
    elt->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting S_figured_bass" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  ++fFiguredBassVoicesCounter;

  std::string parentheses = elt->getAttributeValue ("parentheses");

  fCurrentFiguredBassParenthesesKind =
    msrFiguredBassParenthesesKind::kFiguredBassParenthesesNo; // default value

  if (parentheses.size ()) {
    if (parentheses == "yes")
      fCurrentFiguredBassParenthesesKind =
        msrFiguredBassParenthesesKind::kFiguredBassParenthesesYes;

    else if (parentheses == "no")
     fCurrentFiguredBassParenthesesKind =
        msrFiguredBassParenthesesKind::kFiguredBassParenthesesNo;

    else {
      std::stringstream ss;

      ss <<
        "parentheses value " << parentheses <<
        " should be 'yes' or 'no'";

      musicxmlError (
        gServiceRunData->getInputSourceName (),
        inputLineNumber,
        __FILE__, __LINE__,
        ss.str ());
    }
  }

  fCurrentFiguredBassInputLineNumber   = -1;

  fCurrentFigureNumber = -1;

  fCurrentFigurePrefixKind = msrBassFigurePrefixKind::kBassFigurePrefix_UNKNOWN_;
  fCurrentFigureSuffixKind = msrBassFigureSuffixKind::kBassFigureSuffix_UNKNOWN_;

  fCurrentFiguredBassSoundingWholeNotes = mfRational (0, 1);
  fCurrentFiguredBassDisplayWholeNotes  = mfRational (0, 1);

  fOnGoingFiguredBass = true;
}
```

64.5.2 Second `S_figured_bass` visit

Upon the second visit of `S_figured_bass` , the class `msrFiguredBass` instance is created, populated and appended to `mxsr2msrTranslatorfPendingFiguredBassesList`:

```
void mxsr2msrTranslator::visitEnd ( S_figured_bass& elt )
{
  int inputLineNumber =
    elt->getInputStartLineNumber ();

#ifdef MF_TRACE_IS_ENABLED
  if (gGlobalMxsrOahGroup->getTraceMxsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> End visiting S_figured_bass" <<
      ", line " << inputLineNumber;

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create the figured bass element
#ifdef MF_TRACE_IS_ENABLED
  if (gTraceOahGroup->getTraceFiguredBasses ()) {
    std::stringstream ss;

    ss <<
      "Creating a figured bass" <<
      ", line " << inputLineNumber << ":";

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create the figured bass element
  // if the sounding whole notes is 0/1 (no <duration /> was found), JMI ???
  // it will be set to the next note's sounding whole notes later
  S_msrFiguredBass
    figuredBass =
      msrFiguredBass::create (
        inputLineNumber,
  // JMI      fCurrentPart,
        fCurrentFiguredBassSoundingWholeNotes,
        fCurrentFiguredBassDisplayWholeNotes,
        fCurrentFiguredBassParenthesesKind,
        msrTupletFactor (1, 1));    // will be set upon next note handling

  // attach pending figures to the figured bass element
  if (! fPendingFiguredBassFiguresList.size ()) {
    musicxmlWarning (
      gServiceRunData->getInputSourceName (),
      inputLineNumber,
      "figured-bass has no figures contents, ignoring it");
  }
  else {
    // append the pending figures to the figured bass element
    for (S_msrBassFigure bassFigure : fPendingFiguredBassFiguresList) {
      figuredBass->
        appendFigureToFiguredBass (bassFigure);
    } // for

    // forget about those pending figures
    fPendingFiguredBassFiguresList.clear ();

    // append the figured bass element to the pending figured bass elements list
    fPendingFiguredBassesList.push_back (figuredBass);
  }

  fOnGoingFiguredBass = false;
}
```

64.5.3 Attaching msrFiguredBass instances to notes

msrFiguredBass

The contents of mxsr2msrTranslatorfPendingFiguredBassesList is attached to the class msrNote instance in method
method mxsr2msrTranslator::populateNote ():

```
void mxsr2msrTranslator::populateNote (
  int        inputLineNumber,
  const S_msrNote& newNote)
{
  // ... ... ...

  // handle the pending figured bass elements if any
  if (fPendingFiguredBassesList.size ()) {
    // get voice to insert figured bass elements into
    S_msrVoice
      voiceToInsertFiguredBassesInto =
        fCurrentPart->
          getPartFiguredBassVoice ();

    // ... ... ...

    handlePendingFiguredBasses (
      newNote,
      voiceToInsertFiguredBassesInto);

    // reset figured bass counter
    fFiguredBassVoicesCounter = 0;
  }
}
```

64.5.4 Populating `msrFiguredBass` instances

In src/formats/msr/mxsr2msrTranslator.cpp, the class `msrFiguredBass` instances are populated further and attached to the note by method `mxsr2msrTranslator::handlePendingFiguredBasses ()`:

```
void mxsr2msrTranslator::handlePendingFiguredBasses (
  const S_msrNote&  newNote,
  const S_msrVoice& voiceToInsertInto)
{
  // ... ... ...

  mfRational
    newNoteSoundingWholeNotes =
      fCurrentNote->
        getMeasureElementSoundingWholeNotes (),
    newNoteDisplayWholeNotes =
      fCurrentNote->
        getNoteDisplayWholeNotes ();

  while (fPendingFiguredBassesList.size ()) { // recompute at each iteration
    S_msrFiguredBass
      figuredBass =
        fPendingFiguredBassesList.front ();

    /*
      Figured bass elements take their position from the first
      regular note (not a grace note or chord note) that follows
      in score order. The optional duration element is used to
      indicate changes of figures under a note.
    */

    // set the figured bass element's sounding whole notes
    figuredBass->
      setMeasureElementSoundingWholeNotes (
        newNoteSoundingWholeNotes,
        "handlePendingFiguredBasses()");

    // set the figured bass element's display whole notes JMI useless???
    figuredBass->
      setFiguredBassDisplayWholeNotes (
        newNoteDisplayWholeNotes);

    // set the figured bass element's tuplet factor
    figuredBass->
      setFiguredBassTupletFactor (
        msrTupletFactor (
          fCurrentNoteActualNotes,
          fCurrentNoteNormalNotes));

    // append the figured bass to newNote
    fCurrentNote->
      appendFiguredBassToNote (
        figuredBass);

/* JMI
    // get the figured bass voice for the current voice
    S_msrVoice
      voiceFiguredBassVoice =
        voiceToInsertInto->
          getRegularVoiceForwardLinkToFiguredBassVoice ();

#ifdef MF_SANITY_CHECKS_ARE_ENABLED
  // sanity check
    mfAssert (
      __FILE__, __LINE__,
      voiceFiguredBassVoice != nullptr,
      "voiceFiguredBassVoice is null");
#endif // MF_SANITY_CHECKS_ARE_ENABLED

    // set the figuredBass's voice upLink
    // only now that we know which figured bass voice will contain it
    figuredBass->
      setFiguredBassUpLinkToVoice (
        voiceFiguredBassVoice);

    // append the figured bass to the figured bass voice for the current voice
    voiceFiguredBassVoice->
      appendFiguredBassToVoice (
        figuredBass);
*/

    // don't append the figured bass to the part figured bass voice
    // before the note itself has been appended to the voice

    // remove the figured bass from the list
    fPendingFiguredBassesList.pop_front ();
  } // while
}
```

```
%void mxsr2msrTranslator::copyNoteHarmoniesToChord (
%  S_msrNote note, S_msrChord chord)
%{
%  // copy note's harmony if any from the first note to chord
%
%  const std::list<S_msrHarmony>&
%    noteHarmoniesList =
%      note->getNoteHarmoniesList ();
%
%  if (noteHarmoniesList.size ()) {
%    std::list<S_msrHarmony>::const_iterator i;
%    for (i = noteHarmoniesList.begin (); i != noteHarmoniesList.end (); ++i) {
%      S_msrHarmony harmony = (*i);
%
%#ifdef MF_TRACE_IS_ENABLED
%      if (gTraceOahGroup->getTraceHarmonies ()) {
%        std::stringstream ss;
%
%        ss <<
%          "Copying harmony '" <<
%          harmony->asString () <<
%          "' from note " << note->asString () <<
%          " to chord '" << chord <<
%          '\'' <<
%          std::endl;
%
%      gWaeHandler->waeTraceWithoutInputLocation (
%        __FILE__, __LINE__,
%        ss.str ());
%      }
%#endif // MF_TRACE_IS_ENABLED
%
%      chord->
%        appendHarmonyToChord (harmony);
%
%    } // for
%  }
%}
%
```

64.5.5 Inserting S_msrFiguredBass instances in the part figured bass voice

Method msrVoice::appendNoteToVoice () in src/formats/msr/msrNotes.cpp inserts the figured bass elements in the part figured bass voice:

```
void msrVoice::appendNoteToVoice (const S_msrNote& note)
{
  // ... ... ...

  // are there figured bass elements attached to this note?
  const std::list<S_msrFiguredBass>&
    noteFiguredBassesList =
      note->
        getNoteFiguredBassesList ();

  if (noteFiguredBassesList.size ()) {
    // get the current part's figured bass voice
    S_msrVoice
      partFiguredBassVoice =
        part->
          getPartFiguredBassVoice ();

    for (S_msrFiguredBass figuredBass : noteFiguredBassesList) {
      // append the figured bass element to the part figured bass voice
      partFiguredBassVoice->
        appendFiguredBassToVoice (
          figuredBass);
    } // for
  }
};
```

64.6 Translating figured bass from MSR to MSR

In src/passes/msr2msr/msr2msrTranslator.cpp, a newborn clone of the figured bass element is created upon the first visit, stored in msr2msrTranslatorfCurrentFiguredBassClone, and appended to the current non grace note clone, the current chord clone or to the current voice clone, if the latter is a figured bass voice:

```
void msr2msrTranslator::visitStart (S_msrFiguredBass& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrFiguredBass '" <<
      elt->asString () <<
      '\'' <<
      ", fOnGoingFiguredBassVoice = " << fOnGoingFiguredBassVoice <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create a figured bass element newborn clone
  fCurrentFiguredBassClone =
    elt->
      createFiguredBassNewbornClone (
        fCurrentVoiceClone);

  if (fOnGoingNonGraceNote) {
    // append the figured bass to the current non-grace note clone
    fCurrentNonGraceNoteClone->
      appendFiguredBassToNote (
        fCurrentFiguredBassClone);

    // don't append the figured bass to the part figured bass,  JMI ???
    // this will be done below
  }

  /* JMI
  else if (fOnGoingChord) {
    // register the figured bass in the current chord clone
    fCurrentChordClone->
      setChordFiguredBassesList (fCurrentFiguredBassClone); // JMI
  }
  */

  else if (fOnGoingFiguredBassVoice) { // JMI
    /*
    // register the figured bass in the part clone figured bass
    fCurrentPartClone->
      appendFiguredBassToPartClone (
        fCurrentVoiceClone,
        fCurrentFiguredBassClone);
        */
    // append the figured bass to the current voice clone
    fCurrentVoiceClone->
      appendFiguredBassToVoiceClone (
        fCurrentFiguredBassClone);
  }

  else {
    std::stringstream ss;

    ss <<
      "figured bass is out of context, cannot be handled: " <<
      elt->asShortString ();

    msrInternalError (
      gServiceRunData->getInputSourceName (),
      elt->getInputStartLineNumber (),
      __FILE__, __LINE__,
      ss.str ());
  }
}
```

There are only fields updates upon the second visit:

```
void msr2msrTranslator::visitEnd (S_msrFiguredBass& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> End visiting msrFiguredBass '" <<
      elt->asString () <<
      '\'' <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fCurrentFiguredBassClone = nullptr;
}
```

64.7 Translating figured bass from MSR to LPSR

The same occurs in src/passes/msr2lpsr/msr2lpsrTranslator.cpp: a newborn clone of the figured bass element is created and appended to the current non grace note clone, the current chord clone or to the current voice clone, if the latter is a figured bass voice:

```
void msr2lpsrTranslator::visitStart (S_msrFiguredBass& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrFiguredBass '" <<
      elt->asString () <<
      '\'' <<
      ", fOnGoingFiguredBassVoice = " << fOnGoingFiguredBassVoice <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create a figured bass newborn clone
  fCurrentFiguredBassClone =
    elt->
      createFiguredBassNewbornClone (
        fCurrentVoiceClone);

  if (fOnGoingNonGraceNote) {
    // append the figured bass to the current non-grace note clone
    fCurrentNonGraceNoteClone->
      appendFiguredBassToNote (
        fCurrentFiguredBassClone);

    // don't append the figured bass to the part figured bass,  JMI ???
    // this will be done below
  }

  /* JMI
  else if (fOnGoingChord) {
    // register the figured bass in the current chord clone
    fCurrentChordClone->
      setChordFiguredBassesList (fCurrentFiguredBassClone); // JMI
  }
  */

  else if (fOnGoingFiguredBassVoice) { // JMI
    /*
    // register the figured bass in the part clone figured bass
    fCurrentPartClone->
      appendFiguredBassToPartClone (
        fCurrentVoiceClone,
        fCurrentFiguredBassClone);
        */
    // append the figured bass to the current voice clone
    fCurrentVoiceClone->
      appendFiguredBassToVoiceClone (
        fCurrentFiguredBassClone);
  }

  else {
    std::stringstream ss;

    ss <<
      "figured bass is out of context, cannot be handled: " <<
      elt->asShortString ();

    msrInternalError (
      gServiceRunData->getInputSourceName (),
      elt->getInputStartLineNumber (),
      __FILE__, __LINE__,
      ss.str ());
  }
}
```

Here too, there are only fields updates upon the second visit of `S_msrFiguredBass` instances:

```
void msr2lpsrTranslator::visitEnd (S_msrFiguredBass& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> End visiting msrFiguredBass '" <<
      elt->asString () <<
      '\'' <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  fCurrentFiguredBassClone = nullptr;
}
```

64.8 Translating figured bass from LPSR to LilyPond

This is done in src/passes/lpsr2lilypond/.

There is only one visit of class `msrFiguredBass` instances in src/passes/lpsr2lilypond/lpsr2lilypondTranslator.cpp.

The LilyPond code is generated only if the figured bass element belongs to a figured bass voice: this is where denormalization ends in the workflow:

```
void msr2lpsrTranslator::visitStart (S_msrFiguredBass& elt)
{
#ifdef MF_TRACE_IS_ENABLED
  if (gMsrOahGroup->getTraceMsrVisitors ()) {
    std::stringstream ss;

    ss <<
      "--> Start visiting msrFiguredBass '" <<
      elt->asString () <<
      '\'' <<
      ", fOnGoingFiguredBassVoice = " << fOnGoingFiguredBassVoice <<
      ", line " << elt->getInputStartLineNumber ();

    gWaeHandler->waeTraceWithoutInputLocation (
      __FILE__, __LINE__,
      ss.str ());
  }
#endif // MF_TRACE_IS_ENABLED

  // create a figured bass newborn clone
  fCurrentFiguredBassClone =
    elt->
      createFiguredBassNewbornClone (
        fCurrentVoiceClone);

  if (fOnGoingNonGraceNote) {
    // append the figured bass to the current non-grace note clone
    fCurrentNonGraceNoteClone->
      appendFiguredBassToNote (
        fCurrentFiguredBassClone);

    // don't append the figured bass to the part figured bass,  JMI ???
    // this will be done below
  }

  /* JMI
  else if (fOnGoingChord) {
    // register the figured bass in the current chord clone
    fCurrentChordClone->
      setChordFiguredBassesList (fCurrentFiguredBassClone); // JMI
  }
  */

  else if (fOnGoingFiguredBassVoice) { // JMI
    /*
    // register the figured bass in the part clone figured bass
    fCurrentPartClone->
      appendFiguredBassToPartClone (
        fCurrentVoiceClone,
        fCurrentFiguredBassClone);
        */
    // append the figured bass to the current voice clone
    fCurrentVoiceClone->
      appendFiguredBassToVoiceClone (
        fCurrentFiguredBassClone);
  }

  else {
    std::stringstream ss;

    ss <<
      "figured bass is out of context, cannot be handled: " <<
      elt->asShortString ();

    msrInternalError (
      gServiceRunData->getInputSourceName (),
      elt->getInputStartLineNumber (),
      __FILE__, __LINE__,
      ss.str ());
  }
}
```

Chapter 65

Lyrics handling

Lyrics are presented at section 20.44 [Lyrics], page 216.

65.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

65.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

65.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

65.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

65.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Chapter 66

MIDI handling

MIDI is presented at section 20.45 [MIDI], page 217.

At the day of this writing, MIDI handling is partial, i.e. not all MIDI elements present in MusicXML are incorporated in MSR and no MIDI data can generated generated by MusicFormats.

66.1 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

66.2 Translating from MXSR to MSR (`src/passes/mxsr2msr/`)

66.3 Translating from MSR to MSR (`src/passes/msr2msr/`)

66.4 Translating from MSR to LPSR (`src/passes/msr2lpsr/`)

66.5 Translating from LPSR to LilyPond (`src/passes/lpsr2lilypond/`)

Part XIV

MusicFormats Scripting Language (MFSL)

Chapter 67

MFSL (MusicFormats Scripting Language

67.1 A script example

This script illustrates the basic features of MFSL:

```
#!//Users/jacquesmenu/musicformats-git-dev/build/bin/mfsl

# the MusicFormats service to be used
service : xml2ly

# the input file
input :test.mfsl

# parts
  -keep-musicxml-part-id P1

# the voices choice
choice VOICES_CHOICE : voice1Only | voice2Only ;
  # could be : choice VOICES_CHOICE : ... ... ... ;

set VOICES_CHOICE = voice1Only ;
  # change this to voice2Only to switch to another subset of options
  # could even be parameter to the script such a $1

# choose which options to use according to VOICES_CHOICE
case VOICES_CHOICE :
  voice1Only:
    -title "Joli morceau - voix 1"
    -ignore-msr-voice Part_POne_Staff_One_Voice_Two
  ;

  voice2Only:
    -title "Joli morceau - voix 2"
    --ignore-msr-voice Part_POne_Staff_One_Voice_One

    -display-options-values

    -global-staff-size 25.5
  ;
;
```

This first line of an MFSL script is the so-called *shebang* containing the path to the interpreter, allow for running such scripts by their name provided they are made executable.

67.2 Implementation principles

MFSL is implemented with the `flex` and `bison` C++ code generators:

- `src/interpreters/mfsl/mfslScanner.ll` contains the `flex` lexical description of MFSL.

  It is used to create `src/interpreters/mfsl/mfslScanner.cpp`;
- `src/interpreters/mfsl/mfslParser.yy` is the syntax and semantics description of MFSL.

  From it, `bison` creates `src/interpreters/mfsl/mfslParser.h`, `src/interpreters/mfsl/mfslParser.cpp` and `src/interpreters/mfsl/mfslParser.output`.

  The latter file can be used to check the grammar, in particular if `LR` conflicts are detected;
- communication between the code generated this way is done by a so-called *driver*, along the lines of the `C++-calc` example provided by `bison` v3.8.1;
- the way the tokens description is shared by the scanner and parser is described at section 67.7 [Tokens description], page 387;
- the whole power of OAH is used to handle the contents of MFSL scripts as well as the options to the MFSL interpreter itself.

Only the predefined `bool` type is used, since the generated C++ code relies on this. This is why `getValue ()` is used in `src/clisamples/mfsl.cpp`:

```
  std::string        theMfService;
  std::string        theInputFile;
  oahOptionsAndArguments optionsAndArguments;

  err =
    launchMfslInterpreter (
      inputSourceName,
      traceScanning.getValue (),
      traceParsing.getValue (),
      displayTokens.getValue (),
      displayNonTerminals.getValue (),
      theMfService,
      theInputFile,
      optionsAndArguments);
```

67.3 The contents of the MFSL folder

`src/interpreters/mfsl/location.hh` defines the `yy::location` class, that contains the script file name and input line number:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/interpreters/mfsl > ls -sal
total 776
  0 drwxr-xr-x  28 jacquesmenu  staff    896 Mar 15 05:16 .
  0 drwxr-xr-x@  4 jacquesmenu  staff    128 Mar 13 00:47 ..
 16 -rw-r--r--@  1 jacquesmenu  staff   6148 Mar 14 10:18 .DS_Store
  8 -rw-r--r--@  1 jacquesmenu  staff   1266 Mar 15 05:15 Makefile
 16 -rw-r--r--@  1 jacquesmenu  staff   7864 Mar 15 05:16 location.hh
 24 -rw-r--r--@  1 jacquesmenu  staff  10106 Mar 14 18:26 mfslBasicTypes.cpp
 16 -rw-r--r--@  1 jacquesmenu  staff   4568 Mar 14 18:16 mfslBasicTypes.h
  8 -rw-r--r--@  1 jacquesmenu  staff   1585 Mar 15 05:12 mfslDriver.cpp
  8 -rw-r--r--@  1 jacquesmenu  staff   3413 Mar 15 05:12 mfslDriver.h
  8 -rw-r--r--@  1 jacquesmenu  staff   3041 Mar  9 07:35 mfslInterpreterComponent.cpp
  8 -rw-r--r--@  1 jacquesmenu  staff    661 Mar  9 07:02 mfslInterpreterInterface.h
 24 -rw-r--r--@  1 jacquesmenu  staff  11981 Mar 10 11:38 mfslInterpreterInsiderHandler.
    cpp
 16 -rw-r--r--@  1 jacquesmenu  staff   5270 Mar 10 07:11 mfslInterpreterInsiderHandler.h
  8 -rw-r--r--@  1 jacquesmenu  staff   1161 Mar 15 05:13 mfslInterpreterInterface.h
 16 -rw-r--r--@  1 jacquesmenu  staff   7116 Mar 14 15:53 mfslInterpreterOah.cpp
 16 -rw-r--r--@  1 jacquesmenu  staff   4692 Mar 14 15:51 mfslInterpreterOah.h
 24 -rw-r--r--@  1 jacquesmenu  staff  10070 Mar 14 15:53 mfslInterpreterRegularHandler.
    cpp
  8 -rw-r--r--@  1 jacquesmenu  staff   3533 Mar  9 08:22 mfslInterpreterRegularHandler.h
 88 -rw-r--r--   1 jacquesmenu  staff  43880 Mar 15 05:16 mfslParser.cpp
 96 -rw-r--r--   1 jacquesmenu  staff  45868 Mar 15 05:16 mfslParser.h
 24 -rw-r--r--@  1 jacquesmenu  staff  10722 Mar 13 16:57 mfslParser.output
 16 -rw-r--r--@  1 jacquesmenu  staff   5930 Mar 14 18:19 mfslParser.yy
136 -rw-r--r--   1 jacquesmenu  staff  68514 Mar 15 05:16 mfslScanner.cpp
 24 -rw-r--r--@  1 jacquesmenu  staff  11251 Mar 15 05:12 mfslScanner.ll
144 -rw-r--r--@  1 jacquesmenu  staff  71091 Mar 15 05:14 mfslScanner.log
  8 -rw-r--r--@  1 jacquesmenu  staff   2047 Mar  9 11:45 mfslWae.cpp
  8 -rw-r--r--@  1 jacquesmenu  staff   3681 Mar  9 11:44 mfslWae.h
  8 -rwxr-xr-x@  1 jacquesmenu  staff    817 Mar 14 18:20 test.mfsl
```

67.4 The MFSL basic types

67.5 The MFSL Makefile

This `Makefile` is quite simple: the options to `flex` and `bison` are placed in `src/interpreters/mfsl/mfslScanner.ll` and `src/interpreters/mfsl/mfslParser.yy`, respectively:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/interpreters/mfsl > cat Makefile
# ... ... ...

# variables
# ------------------------------------------------------------------------------

MAKEFILE = Makefile

GENERATED_FILES  = mfslParser.h mfslScanner.cpp mfslParser.cpp

BISON = bison
FLEX  = flex

CXXFLAGS = -I.. -DMAIN


# implicit target
# ------------------------------------------------------------------------------

all : $(GENERATED_FILES)


# generation rules
# ------------------------------------------------------------------------------

mfslScanner.cpp : $(MAKEFILE) mfslScanner.ll
  $(FLEX) -omfslScanner.cpp mfslScanner.ll


mfslParser.h mfslParser.cpp : $(MAKEFILE) mfslParser.yy
  $(BISON) --defines=mfslParser.h -obj mfslParser.cpp mfslParser.yy


# clean
# ------------------------------------------------------------------------------
```

```

clean:
  rm -f $(GENERATED_FILES)
```

67.6 Locations handling

67.7 Tokens description

The tokens are described in src/interpreters/mfsl/mfslParser.yy, such as:

```
%token <std::string> OPTION "option"
```

Both OPTION and "option" can be used in the productions, but the grammar is more readable if the capitalized name is used:

```
Option
  : OPTION
    {
      if (drv.getDisplayNonTerminals ()) {
        gLog <<
          "  ==> option " << $1 <<
          std::endl << std::endl;
      }

      $$ = oahOptionNameAndValue::create ($1, "");
    }

  | OPTION OptionValue
    {
      if (drv.getDisplayNonTerminals ()) {
        gLog <<
          "  ==> option " << $1 << ' ' << $2 <<
          std::endl << std::endl;
      }

      $$ = oahOptionNameAndValue::create ($1, $2);
    }
;
```

In case of error, "option" is used to display a message to the user.

The name OPTION is used in src/interpreters/mfsl/mfslScanner.ll prefixed by yy::parser::make_:

```
"--"{name} |
"-"{name} {
  if (drv.getTraceTokens ()) {
    gLog << "--> " << drv.getScannerLocation () <<
    ": option [" << yytext << ']' <<
    std::endl;
  }
  return yy::parser::make_OPTION (yytext, loc);
}
```

The suffix after make_ has to be defined in the src/interpreters/mfsl/mfslParser.yy for this to do the link between the Flex-generated and Bison-generated code:

```
%token <std::string> OPTION "option"
```

In src/interpreters/mfsl/mfslParser.cpp, this becomes:

```
      case symbol_kind::S_OPTION: // "option"
```

We don't have to create method `yy::parser::make_OPTION ()` ourselves, though: it is taken care of by Bison itself, since it returns a type `char*`.

The `calc++` example in the `bison` documentation contains the case of numbers:

```
%{
  // A number symbol corresponding to the value in S.
  yy::parser::symbol_type
  make_NUMBER (const std::string &s, const yy::parser::location_type& loc);
%}

// ... ... ...

yy::parser::symbol_type
make_NUMBER (const std::string &s, const yy::parser::location_type& loc)
{
  errno = 0;
  long n = strtol (s.c_str(), NULL, 10);
  if (! (INT_MIN <= n && n <= INT_MAX && errno != ERANGE))
    throw yy::parser::syntax_error (loc, "integer is out of range: " + s);
  return yy::parser::make_NUMBER ((int) n, loc);
}
```

67.8 The driver

Class `mfslDriver` contains everything needed to let the code generated by `flex` and `bison` communicate with each other, as well as any work variables needed during the analysis of MFSL input. This latter point allows for multiple analyzers to coexist.

`src/interpreters/mfsl/mfslDriver.h` contains a prototype of function `yylex ()`:

```
//______________________________________________________________________________
// Give Flex the prototype of yylex we want ...
# define YY_DECL \
  yy::parser::symbol_type yylex (mfslDriver& drv)
// ... and declare it for the parser's sake.
YY_DECL;
```

Then it contains the declaration of class `mfslDriver`:

```
// Conducting the whole scanning and parsing of MFSL
class   mfslDriver
{
  public:

    // constants
    // ------------------------------------------------------

    static const std::string   K_ALL_PSEUDO_LABEL_NAME;

    // // constructor/destructor
    // ------------------------------------------------------

                          mfslDriver ();

    virtual               ~mfslDriver ();

  // ... ... ...

  public:

    // public services
    // ------------------------------------------------------

    // run the parser, return 0 on success
    int                     parseInput_Pass1 ();

    // handling the scanner
    void                    scanBegin ();
    void                    scanEnd ();

  // ... ... ...

  private:

    // private fields
    // ------------------------------------------------------

    // the name of the MusicFormats service
    std::string             fService;

    // the name of the MusicFormats script
    std::string             fScriptName;

    // the names of the input sources
    std::list<std::string>
                            fInputSoucesList;


    // scanning
    bool                    fTraceScanning;
    mfsl::location          fScannerLocation;

  // ... ... ...
};
```

The definitions are placed in two files due to the specificity of the sharing of variables and function in the `flex` and `bison`-generated code:

- src/interpreters/mfsl/mfslDriver.cpp contains method `mfslDriver::parseInput_Pass1 ()`, that runs the parser:

```
int mfslDriver::parseInput_Pass1 ()
{
  // initialize scanner location
  fScannerLocation.initialize (
    &fScriptName);

  // begin scan
  scanBegin ();

  if (fScriptName.empty () || fScriptName == "-") {
    fScriptName = "stdin"; // nicer for warning and error messages
  }

  // do the parsing
  mfsl::parser theParser (*this);

  theParser.set_debug_level (
    fTraceParsing);

  int parseResult = theParser ();

  // end scan
  scanEnd ();

  // ... ... ...

  // do the final semantics check
  finalSemanticsCheck ();

  return parseResult;
}
```

- the remaining code is placed in the third part (service code) of `src/interpreters/mfsl/mfslScanner.ll`, since it needs to access variables in the code generated by `flex`:

```
void mfslDriver::scanBegin ()
{
  yy_flex_debug = fTraceScanning;

  if (fScriptName.empty () || fScriptName == "-") {
    yyin = stdin;
  }

  else if (!(yyin = fopen (fScriptName.c_str (), "r")))
    {
      std::stringstream ss;

      char*
        errorCString =
          mfStrErrorCString ();

      if (errorString != nullptr) {
        ss <<
          gLanguage->cannotOpenScriptForWriting (fScriptName) <<
          ": " <<
          errorString <<
          std::endl;

        mfslFileError (
          fScriptName,
          ss.str ());
      }
    }
}

void mfslDriver::scanEnd ()
{
  fclose (yyin);
}
```

67.9 Lexical analysis

The lexical definition of MFSL in `src/interpreters/mfsl/mfslScanner.ll` is described below.

67.9.1 Flex options

The `prefix` is used to allow for multiple `flex`-generated analyzers to coexist:

```
%option prefix="mfsl"

%option yylineno

%option noyywrap
```

```

%option nounput noinput debug interactive
```

67.9.2 Flex regular expressions

The basic ones are:

```
blank                          [ \t\r]
endOfLine                      [\n]
character                      .

letter                         [A-Za-zéèêàâôùûî]
digit                          [0-9]

name                           {letter}(_|-|\.|{letter}|{digit})*
integer                        {digit}+
exponent                       [eE][+-]?{integer}

singleleQuote                  [']
doubleQuote                    ["]
tabulator                      [\t]
backSlash                      [\\]

... ... ...

%{
  // Code run each time a pattern is matched.
  # define YY_USER_ACTION  loc.columns (yyleng);
%}
```

Some exclusive modes are used for strings and comments:

```
%x                             SINGLE_QUOTED_STRING_MODE
%x                             DOUBLE_QUOTED_STRING_MODE

%x                             COMMENT_TO_END_OF_LINE_MODE
%x                             PARENTHESIZED_COMMENT_MODE
```

Strings must be stored in a private buffer:

```
/* strings */

#define                        STRING_BUFFER_SIZE 1024
char                           pStringBuffer [STRING_BUFFER_SIZE];

// A handy shortcut to the location held by the mfslDriver
mfsl::location& loc = drv.getScannerLocationNonConst ();
```

Locating the tokens in the the MFSL input text is done with:

```
// Code run each time yylex() is called
loc.step ();
```

This lead for example to:

```
{blank} {
  loc.step ();
}

{endOfLine} {
  loc.lines (yyleng); loc.step ();
}
```

The numbers are handled by:

```
{integer}"."{integer}({exponent})? |
{integer}{exponent} {
  if (drv.getTraceTokens ()) {
    gLog <<
      "--> " << drv.getScannerLocation () <<
      " double: " << yytext <<
      std::endl;
  }
  return yy::parser::make_DOUBLE (yytext, loc);
}

{integer} {
  if (drv.getTraceTokens ()) {
    gLog <<
      "--> " << drv.getScannerLocation () <<
      " integer: " << yytext <<
      std::endl;
  }
  return yy::parser::make_INTEGER (yytext, loc);
}
```

The MFSL keywords are handled with the `make_...` facility:

```
"service" {
  if (drv.getTraceTokens ()) {
    gLog <<
      "--> " << drv.getScannerLocation () << ": " << yytext <<
      std::endl;
  }
  return yy::parser::make_SERVICE (loc);
}
```

The names and the options are handled by:

```
{name} {
  if (drv.getDisplayTokens ()) {
    gLog << "--> " << drv.getScannerLocation () <<
    ": name [" << yytext << ']' <<
    std::endl;
  }

  loc.begin.column += yyleng;
  loc.step ();

  return
    mfsl::parser::make_NAME (yytext, loc);
}



"--"{name} |
"-"{name} {
  if (drv.getTraceTokens ()) {
    gLog << "--> " << drv.getScannerLocation () <<
    ": option [" << yytext << ']' <<
    std::endl;
  }
  return yy::parser::make_OPTION (yytext, loc);
}



"(" {
  if (drv.getTraceTokens ()) {
    gLog <<
      "--> " << drv.getScannerLocation () << ": " << yytext <<
      std::endl;
  }
  return yy::parser::make_LEFT_PARENTHESIS (loc);
}
```

The catchall rule issues an error message:

```
. {
   throw mfsl::parser::syntax_error (
     loc,
     "### invalid character: " + std::string (yytext));
}
```

And the end of the MFSL input is handled this way:

```
<<EOF>> {
  return
    mfsl::parser::make_YYEOF (loc);
}
```

67.10 Syntax and semantic analysis

67.10.1 Bison options for MFSL

Setting api.prefix allows for multiple analyzers to coexist:

```
%skeleton "lalr1.cc" // -*- C++ -*-
%require "3.8.1"
%defines

%define api.prefix {mfsl}

%define api.token.raw

%define api.token.constructor
%define api.value.type variant
%define parse.assert

%code requires {
  #include <string>

  class   mfslDriver;
}

// the parsing context
%param { mfslDriver& drv } // declaration, any parameter name is fine

%verbose // to produce mfslParser.output

%locations

// other Bison options
%define parse.trace
%define parse.error detailed
%define parse.lac full
// %define api.pure full

%printer { yyo << $$; } <*>;


```

```
%code {
  #include "mfslBasicTypes.h"
}
```

67.10.2 The MFSL tokens

The MFSL tokens are:

```
%define api.token.prefix {MFSL_TOK_}

%token
  BAR          "|"
  AMPERSAND    "&"
  EQUAL         "="
  SEMICOLON    ";"
  COLON        ":"
  COMMA        ","

  SERVICE      "service"
  INPUT        "input"

  CHOICE       "choice"
  DEFAULT      "default"

  CASE         "case"

  SELECT       "select"
  ALL          "all"
;

%code {
  #include "mfslDriver.h"
}

%token <std::string> INTEGER "integer number"
%token <std::string> DOUBLE  "double number"

%token <std::string> SINGLE_QUOTED_STRING "single quoted_string"
%token <std::string> DOUBLE_QUOTED_STRING "double quoted_string"

%token <std::string> NAME "name"

%token <std::string> OPTION "option"
```

67.10.3 The MFSL non-terminals and axiom

They are:

```
// the MFSL non-terminals
//--------------------------------------------------------------------------------

%nterm <std::string> Number

%nterm <std::string> SingleString
%nterm <std::string> std::string

%nterm <std::string> OptionValue

%nterm <std::string> LabelName


// the MFSL axiom
```

```
//-----------------------------------------------------------------------------

%start Script
```

67.11 Interface to the MFSL parser

This is provided by src/interpreters/mfsl/mfslInterpreterInterface.h:

```
EXP extern mfMusicformatsErrorKind launchMfslInterpreter (
  const std::string&      inputSourceName,
  bool                    traceScanning,
  bool                    traceParsing,
  bool                    displayTokens,
  bool                    displayNonTerminals,
  std::string&                 theMfService,
  std::string&                 theInputFile,
  oahOptionsAndArguments& optionsAndArguments);
```

The definition of this function is placed in src/interpreters/mfsl/mfslScanner.ll:

```
EXP mfMusicformatsErrorKind launchMfslInterpreter (
  const std::string&      inputSourceName,
  bool                    traceScanning,
  bool                    traceParsing,
  bool                    displayTokens,
  bool                    displayNonTerminals,
  std::string&                 theMfService,
  std::string&                 theInputFile,
  oahOptionsAndArguments& optionsAndArguments)
{
  mfMusicformatsErrorKind
    result =
      mfMusicformatsErrorKind::kMusicformatsError_NONE;

  mfslDriver
    theDriver (
      traceScanning,
      traceParsing,
      displayTokens,
      displayNonTerminals);

  int parseResult =
    theDriver.parseFile (inputSourceName);

  gLog <<
    "--> parseResult: " << parseResult <<
    std::endl;

  if (! parseResult) {
    result =
      mfMusicformatsErrorKind::kMusicformatsErrorInvalidFile;
  }

  gLog <<
    "inputFileName: " << theDriver.getInputFileName () <<
    std::endl <<
    "serviceName: " << theDriver.getServiceName () <<
    std::endl;

  theMfService    = theDriver.getServiceName ();
  theInputFile = theDriver.getInputFileName ();

  return result;
}
```

67.12 Running the example MFSL script

Let's show show the MFSL interpreter uses the options above:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/interpreters/mfsl > ./test.mfsl -display-
    tokens -display-non-terminals -display-options-values
  The options values for //Users/jacquesmenu/musicformats-git-dev/build/bin/mfsl are:
    MFSL group (-help-mfsl-group, -hmfsl-group), 2 atoms selected:
    --------------------------
      MFSL (-help-mfsl, -hmfsl), 2 atoms selected:
        fTraceTokens                      : true, selected option
        fDisplayNonTerminals                : true, selected option

    Options and help group (-help-oah-group, -hoah-group), 1 atom selected:
    --------------------------
      Options and help (-help-oah, -hoah), 1 atom selected:
        fDisplayOptionsValues               : true, selected option


--> ./test.mfsl:2.1-39: service
--> ./test.mfsl:2.41: :
--> ./test.mfsl:2.43-48: name [xml2ly]
  ==> service: xml2ly

--> ./test.mfsl:4.1-22: input
--> ./test.mfsl:4.24: :
--> ./test.mfsl:4.25-33: name [test.mfsl]
  ==> input: test.mfsl

--> test.mfsl:6.11-32: option [-keep-musicxml-part-id]
--> test.mfsl:6.34-35: name [P1]
  ==> option -keep-musicxml-part-id P1

--> test.mfsl:8.1-26: choice
--> test.mfsl:8.28-40: name [VOICES_CHOICE]
--> test.mfsl:8.42: :
--> test.mfsl:8.44-53: name [voice1Only]
--> test.mfsl:8.55: |
--> test.mfsl:8.57-66: name [voice2Only]
--> test.mfsl:8.68: ;
  ==> ChoiceDeclaration VOICES_CHOICE : ...

--> test.mfsl:10.1-3: set
--> test.mfsl:10.5-17: name [VOICES_CHOICE]
--> test.mfsl:10.19: =
--> test.mfsl:10.21-30: name [voice1Only]
--> test.mfsl:10.32: ;
  ==> ChoiceSetting, set VOICES_CHOICE = voice1Only

--> test.mfsl:12.1-61: case
--> test.mfsl:12.63-75: name [VOICES_CHOICE]
--> test.mfsl:12.77: :
--> test.mfsl:13.2-11: name [voice1Only]
--> test.mfsl:13.12: :
--> test.mfsl:14.5-10: option [-title]
--> test.mfsl:14.12-34: double quoted std::string ["]
  ==> option -title "

--> test.mfsl:15.5-21: option [-ignore-msr-voice]
--> test.mfsl:15.23-51: name [Part_POne_Staff_One_Voice_Two]
  ==> option -ignore-msr-voice Part_POne_Staff_One_Voice_Two

--> test.mfsl:16.3: ;
  ==> Case voice1Only : ...

--> test.mfsl:18.2-11: name [voice2Only]
--> test.mfsl:18.12: :
--> test.mfsl:19.5-10: option [-title]
--> test.mfsl:19.12-34: double quoted std::string ["]
  ==> option -title "

--> test.mfsl:20.5-22: option [--ignore-msr-voice]
--> test.mfsl:20.24-52: name [Part_POne_Staff_One_Voice_One]
  ==> option --ignore-msr-voice Part_POne_Staff_One_Voice_One

--> test.mfsl:22.5-27: option [-display-options-values]
--> test.mfsl:24.5-22: option [-global-staff-size]
  ==> option -display-options-values

--> test.mfsl:24.24-27 double: 25.5
  ==> option -global-staff-size 25.5

--> test.mfsl:25.3: ;
  ==> Case voice2Only : ...

--> test.mfsl:26.1: ;
  ==> CaseStatement, VOICES_CHOICE : ...

--> parseResult: 0
==> inputFileName: test.mfsl
==> serviceName:      xml2ly
jacquesmenu@macmini: ~/musicformats-git-dev/src/interpreters/mfsl >
```

67.12.1 Error recovery

The MFSL interpreter uses a variant of the *stopper sets* method that was present in the early Pascal and Pascal-S converters. The latter passed a set of tokens not to be overtaken to the procedures in charge of accepting the various statements in the language. Strangely enough, this was not done for declarations.

We use a stack of tokens sets that grows and shrinks in parallel with the accepting functions, to know more contextual informations when deciding wether to consume a token or not. The corresponding term is it shift when building the analysis tables in LR technology.

Part XV

Music Scores Description Language (MSDL)

Chapter 68

MSDL (Music Scores Description Language

MSDL is an attempt at a description of music score in a non-linear way, much like a painter puts touches of paint on his work. This is also what users do with GUI music scoring applications, but scores textual descriptions such as LilyPond and Guido impose a linear, left to right, writing of the scores contents.

Contrary to LilyPond, the `|` token in MSDL is not the end of a measure. Writing `|2` means that the music that follows will be placed in a new layer in measure 2.

68.1 Main features of MSDL

They are:

- note are written much like in LilyPond such as `b2...`;
- the keywords such as `pitches` and `music`, are reserved;
- they are available in a number of languages such as english, french, german and italian. It is easy to add other languages;

A first, limited converter is provided by MusicFormats with service `msdl`. It also performs reserved keywords translation from one language to another:

68.2 MSDL basic types

Some types used thoughout MSR are defined in src/formats/msdl/msdlEnumTypes.h/.cpp:

```
jacquesmenu@macmini: ~/musicformats-git-dev/src/formats/msdl > egrep -rIn '^// ' msdlEnumTypes.h
msdlEnumTypes.h:28:// user languages
msdlEnumTypes.h:52:// comments types
msdlEnumTypes.h:74:// initialization
```

68.3 What the MSDL converter does

```
jacquesmenu@macmini: ~/musicformats-git-dev/msdl > msdl -about
What msdlConverter does:

    This multi-pass converter performs various passes depending on the output generated,
    which should be specified a '-lilypond', '-braille', '-musicxml' or '-guido' option.

    Other passes are performed according to the options, such as
    displaying views of the internal data or printing a summary of the score.

    The activity log and warning/error messages go to standard error.

The output format is selected via options.
```

68.3.1 LilyPond generation

```
jacquesmenu@macmini: ~/musicformats-git-dev/msdl > msdl -lilypond -about
What msdlConverter does:

    This multi-pass converter basically performs 3 passes when generating LilyPond output
    output:

        Pass 1:  converts the MSDL input into a first MSR
        Pass 2:  converts the first MSR into a second MSR;
        Pass 3:  converts the second MSR into a
                 LilyPond Score Representation (LPSR);
        Pass 4:  converts the LPSR to LilyPond code
                 and writes it to standard output.

    Other passes are performed according to the options, such as
    displaying views of the internal data or printing a summary of the score.

    The activity log and warning/error messages go to standard error.
```

68.3.2 Braille generation

```
jacquesmenu@macmini: ~/musicformats-git-dev/msdl > msdl -braille -about
What msdlConverter does:

    This multi-pass converter basically performs 4 passes when generating braille output
    output:

        Pass 1:  converts the MSDL input into a first MSR
        Pass 2:  converts the first MSR into a second MSR;
        Pass 3a: converts the second MSR into a
                 Braille Score Representation (BSR)
                 containing one Braille page per MusicXML page;
        Pass 3b: converts the BSRinto another BSR
                 with as many Braille pages as needed
                 to fit the line and page lengthes;
        Pass 4:  converts the BSR to Braille text
                 and writes it to standard output.)

    In this preliminary version, pass 2b merely clones the BSR it receives.

    Other passes are performed according to the options, such as
    displaying views of the internal data or printing a summary of the score.

    The activity log and warning/error messages go to standard error.
```

68.3.3 MusicXML generation

```
jacquesmenu@macmini: ~/musicformats-git-dev/msdl > msdl -musicxml -about
What msdlConverter does:

    This multi-pass converter basically performs 4 passes when generating MusicXML output
   output:

        Pass 1:  converts the MSDL input into a first MSR
        Pass 2:  converts the first MSR into a second MSR;
        Pass 3:  converts the second MSR into an MusicXML tree;
        Pass 4:  converts the MusicXML tree to MusicXML code
                 and writes it to standard output.

    Other passes are performed according to the options, such as
    displaying views of the internal data or printing a summary of the score.

    The activity log and warning/error messages go to standard error.
```

68.3.4 Guido generation

```
jacquesmenu@macmini: ~/musicformats-git-dev/msdl > msdl -guido -about
What msdlConverter does:

    This multi-pass converter basically performs 4 passes when generating Guido output
   output:

        Pass 1:  converts the MSDL input into a first MSR
        Pass 2:  converts the first MSR into a second MSR;
        Pass 3:  converts the second MSR into an MusicXML tree;
        Pass 4:  converts the MusicXML tree to Guido code
                 and writes it to standard output.

    Other passes are performed according to the options, such as
    displaying views of the internal data or printing a summary of the score.

    The activity log and warning/error messages go to standard error.
jacquesmenu@macmini: ~/musicformats-git-dev/msdl >
```

68.4 A first example

`HelloWorld.msdl` is a minimal example:

```
%{
  The unavoidable HelloWorld score
%}


% the language used for the pitches
% -----------------------------------------------

pitches english             % default is english


% is there an anacrusis?
% -----------------------------------------------

anacrusis                  % measure numbers start at 0


% the structure
% ----------------------------------------------

% score helloWorld = "Hello World in MSPL" {

  music {
    |1 c2.. d''8
    |||                     % final bar
  }

% } % helloWorld
```

68.5 First example output from the MSDL converter

Compiling `HelloWorld.msdl` to LilyPond, we get the output below.

68.5.1 LilyPond output

```
\version "2.24.0"

% Comment or adapt next line as needed (default is 20)
#(set-global-staff-size 20 )

% Pick your choice from the next two lines as needed
%myBreak = { \break }
myBreak = {}

% Pick your choice from the next two lines as needed
%myPageBreak = { \pageBreak }
myPageBreak = {}

\header {
  title                       = ""
  encodingDate                = "Sunday 2021-05-30 @ 12:11:50 CEST"
  software                    = "MSDL converter 1.0"
}

\paper {
}

\layout {
  \context {
    \Score
    autoBeaming = ##f % to display tuplets brackets
  }
  \context {
    \Voice
  }
}

Part_Part_One_Staff_One_Voice_One = \absolute {
  \language "nederlands"
c2.. d''8 }

\book {
  \score {
    <<

      \new Staff = "Part_Part_One_Staff_One"
      \with {
      }
      <<
```

```
        \context Voice = "Part_Part_One_Staff_One_Voice_One" <<
          \Part_Part_One_Staff_One_Voice_One
        >>
      >>

    >>

    \layout {
      \context {
        \Score
        autoBeaming = ##f % to display tuplets brackets
      }
      \context {
        \Voice
      }
    }

    \midi {
      \tempo 4 = 90
    }
  }

}
```

68.5.2 Braille output

With:

```
msdl -braille HelloWorld.msdl -use-encoding-in-file-name -braille-output-kind utf8d
```

we get in file `HelloWorld.msdl_UTF8Debug.brf` Braille 6-dots cells, which can be displayed in a suitable editor as:

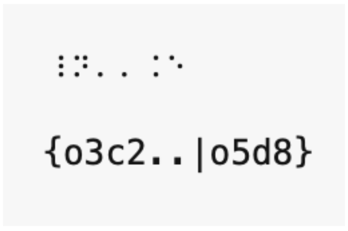


Figure 68.1: Braille for HelloWorld.xml with interpretation

The interpretation shows a textual view of the contents of the previous line. `obj*` indicates the octave number.

68.5.3 MusicXML output

Compiling `HelloWorld.msdl` to MusicXML, we get:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 3.1 Partwise//EN"
      "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.1">
    <!-- ===== Created by msdl 0.02 on Sunday 2021-05-30 @ 12:15:44 CEST from HelloWorld.msdl ===== -->
    <work>
        <work-number/>
        <work-title/>
    </work>
    <movement-number/>
    <movement-title/>
    <identification>
        <encoding>
            <software>msdl 0.02, https://github.com/jacques-menu/musicformats</software>
            <encoding-date>2021-05-30</encoding-date>
        </encoding>
        <miscellaneous>
            <miscellaneous-field name="description"/>
        </miscellaneous>
    </identification>
    <part-std::list>
        <score-part id="Part_One">
            <part-name/>
            <score-instrument id="Part_OneI1">
                <instrument-name/>
            </score-instrument>
        </score-part>
    </part-std::list>
    <part id="Part_One">
        <measure number="1">
            <attributes>
                <divisions>2</divisions>
            </attributes>
            <note>
                <pitch>
                    <step>C</step>
                    <octave>3</octave>
                </pitch>
                <duration>7</duration>
                <voice>1</voice>
                <type>half</type>
                <dot/>
                <dot/>
                <staff>1</staff>
            </note>
            <note>
                <pitch>
                    <step>D</step>
                    <octave>5</octave>
                </pitch>
                <duration>1</duration>
                <voice>1</voice>
                <type>eighth</type>
                <staff>1</staff>
            </note>
        </measure>
    </part>
</score-partwise>
```

68.5.4 Guido output

Compiling HelloWorld.msdl to Guido, we get:

```
{[ \staff<1> \set<autoHideTiedAccidentals="on"> \title<""> \barFormat<style= "system",
   range="1"> \bar<hidden="true"> \beamsOff c0/2.. \beamsOff d2/8 ]
  }
```

68.6 A more realistic example

Thanks to Jean Abou-Samra for providing `UnPetitAir.msdl`:

```
%{
  An explicit and implicit voices piano score
%}


% l'identification
% ------------------------------------------------

titre       "Un petit air"
compositeur "Jean Abou Samra"


% la langue pour les hauteurs de notes
% ------------------------------------------------

hauteurs francais         % par défaut: english


% la partition
% ------------------------------------------------

musique unPetitAir =
{
  |1  clef treble
      key c
      time 9/8
      r4. a,4-> <e g bf>8~ <e g bf>4.~

  |2  <e g bf>4. r2.

  %  Maintenant, je reviens en arrière pour la voix supérieure.
  |2  fs''16 gs'' fs''8 cs'' ds'' e'' b' d'' a' e'

  %  La voix inférieure s'éteint.
  |3  c''8 gs' d' c' fs' a' b' gs' b
  |4  a'8 e' a g as gs' d'( a ds)
  |5  e8( b g d' a' e'' b'' c''' b''
  |6  e'''4.) % Rien à la fin.

  %  Je décide d'ajouter une tenue de la basse.
  |5  e2.~ e4.

  %  J'ajoute encore une voix. Au passage, je change la métrique.
  |6  time 6/8
  |6  r8 e'( f') e' c'' d''

  %  Et encore un changement de métrique.
  |7  time 4/4
  |7  e''1~

  %  Je finis la phrase.
  |7  e''4 e' d''8 c'' b' a'
  |8  b'1

  %  Je retourne sur mes pas pour introduire l'ostinato.
  |7  r8 e8 f e f e c' a
  |8  r8 e8 f e c' a e f
  |9  r8 ds e ds e ds b fs

  % etc.
}
```

Jean also provided the output created by hand with LilyPond, see figure 68.2 [Un Petit Air, par Jean Abou-Samra], page 406:

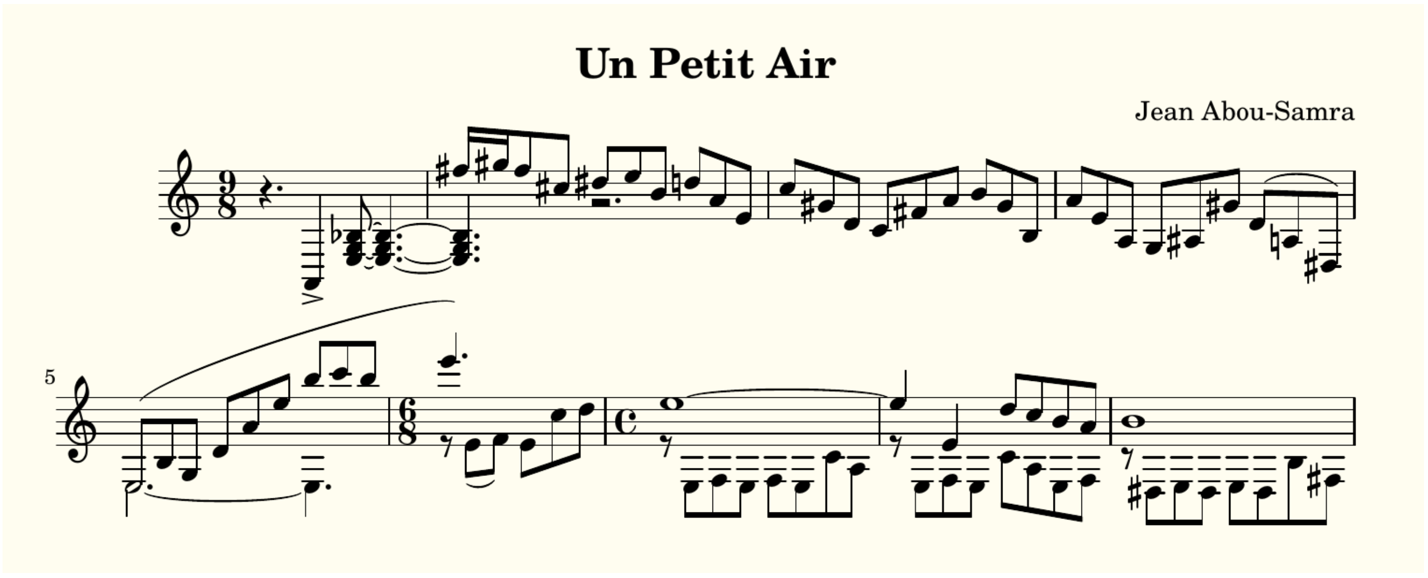


Figure 68.2: Un Petit Air, par Jean Abou-Samra

68.7 Multi-language support

68.7.1 Multi-language messages handling

68.7.2 Multi-language keywords handling

68.8 Lexical analysis

68.9 Music Scores Descriptions Representation (MSDR)

68.10 Syntax and semantic analysis

The language-dependent keywords leads to a recursive descent parser, since `flex`-generated scanners need 'fixed' keyword in the language description.

68.10.1 Error recovery

The MSDL converter uses a variant of the *stopper sets* method that was present in the early Pascal and Pascal-S converters. The latter passed a set of tokens not to be overtaken to the procedures in charge of accepting the various statements in the language. Strangely enough, this was not done for declarations.

We use a stack of tokens sets that grows and shrinks in parallel with the accepting functions, to know more contextual informations when deciding wether to consume a token or not. The corresponding term is it shift when building the analysis tables in LR technology.

Part XVI

Debugging

Chapter 69

Debugging

Debugging MusicFormats can be quite time-consuming. The trace options available have been designed to provide fine-grained trace information to help locate issues.

Functions `catchSignals ()` at the beginning of the `main ()` function can be commented out in order to run a service under a debugger.

File `src/mflibrarymfPreprocessorSettings.h` contains:

```
/*
  MusicFormats Library
  Copyright (C) Jacques Menu 2016-2023

  This Source Code Form is subject to the terms of the Mozilla Public
  License, v. 2.0. If a copy of the MPL was not distributed with this
  file, you can obtain one at http://mozilla.org/MPL/2.0/.

  https://github.com/jacques-menu/musicformats
*/

#ifndef ___mfPreprocessorSettings___
#define ___mfPreprocessorSettings___

/*
  This file groups several build-time setting that influence
  code security and speed, as well as a couple of facilities
*/

//______________________________________________________________________________
// uncomment the following definition if no exceptions display is desired
#define MF_CAUGHT_EXCEPTIONS_DISPLAY_IS_ENABLED

//______________________________________________________________________________
// comment the following definition if abort on internal errors is desired
// CAUTION: DON'T USE THIS IN PRODUCTION CODE,
// since that could kill a session on a web server, for example
#define MF_ABORT_TO_DEBUG_ERRORS_IS_ENABLED

//______________________________________________________________________________
// uncomment the following definition if abort on internal errors is desired
#define MF_SANITY_CHECKS_ARE_ENABLED

//______________________________________________________________________________
// uncomment the following definition if harmonies extra options are desired
#define MF_HARMONIES_EXTRA_IS_ENABLED

  #include "oahHarmoniesExtraOah.h"

//______________________________________________________________________________
```

```
// comment the following definition if no trace is desired
#define MF_TRACE_IS_ENABLED

#ifdef MF_TRACE_IS_ENABLED
  #include "mfTraceOah.h"
#endif // MF_TRACE_IS_ENABLED


#endif // ___mfPreprocessorSettings___
```

69.1 Useful options

Here are the most basing options used when debugging:

- option `-trace-passes, -tpasses` this is the first option to use, to locate in which pass the problem arises
- option `-input-line-numbers, -iln` this option produces the music elements input-line numbers in the output files
- the `-display*` options

69.2 Removing the results of a build

The contents of `distrib/` after a build is:

```
jacquesmenu@macmini: ~/musicformats-git-dev > ll build/
total 80
 0 drwxr-xr-x  10 jacquesmenu  staff    320 May  2 18:18:50 2022 ./
 0 drwxr-xr-x  39 jacquesmenu  staff   1248 May  2 18:18:51 2022 ../
16 -rw-r--r--@  1 jacquesmenu  staff   6148 May  2 15:45:10 2022 .DS_Store
 8 -rw-r--r--@  1 jacquesmenu  staff   1815 Jun 29 09:10:50 2021 Building.md
32 -rw-r--r--   1 jacquesmenu  staff  14849 May  2 18:18:50 2022 CMakeLists.txt
 8 -rw-r--r--   1 jacquesmenu  staff    291 Jun 29 09:10:50 2021 MakePkg.bat
16 -rw-r--r--   1 jacquesmenu  staff   7463 May  2 18:18:50 2022 Makefile
 0 drwxr-xr-x@ 27 jacquesmenu  staff    864 May  2 18:21:17 2022 bin/
 0 drwxr-xr-x@  7 jacquesmenu  staff    224 May  2 18:21:11 2022 lib/
 0 drwxr-xr-x  12 jacquesmenu  staff    384 May  2 18:19:25 2022 libdir/
```

The built files are in `distrib/bin`, `distrib/lib` and `distrib/libdir`. There is no `clean` target in `Makefile`. They can be removed in a single step with this alias:

```
jacquesmenu@macmini: ~/musicformats-git-dev > type rmbuild
rmbuild is aliased to ‘cd ${MUSIC_FORMATS_DEV}/build ; rm -r bin lib libdir; ls -sal’
```

69.3 Reverting to a previous MusicFormats version

The GitHub MusicFormats repository keeps a number of recents releases, such as:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git branch
* master
  v0.9.60
  v0.9.61
  v0.9.62
```

Then `master` branch contains the development version. To switch back to another version, one should check it out:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git checkout v0.9.61
Switched to branch 'v0.9.61'
Your branch is up to date with 'master/v0.9.61'.
jacquesmenu@macmini: ~/musicformats-git-dev > git branch
  master
  v0.9.60
* v0.9.61
  v0.9.62
```

Now building this version with `make`, we get:

```
jacquesmenu@macmini: ~/musicformats-git-dev/build > xml2ly -version
Command line version of musicxml2lilypond converter v0.9.61 (March 3, 2022)
```

Chapter 70

Locating a bug with Git's bisection

A bug appeared in v0.9.63, in which the `-display-lpsr-short, -dlpsrshort` option causes `xml2ly` to crash. The symptom varies with the operating system, pointing to a probable memory corruption.

Reverting to MusicFormats previous versions show that:

- v0.9.60 behaves alright;
- v0.9.61 exhibits the bug:

```
jacquesmenu@macmini: ~/Desktop > lldb -- xml2ly  fullbarrests/MeasureRestWithoutBarLine.xml -display-lpsr-short -aofn
(lldb) target create "xml2ly"
Current executable set to 'xml2ly' (x86_64).
(lldb) settings set -- target.run-args  "fullbarrests/MeasureRestWithoutBarLine.xml" "-display-lpsr" "-aofn"
(lldb) r
Process 28676 launched: '/Users/jacquesmenu/musicformats-git-dev/build/bin/xml2ly' (x86_64)
        The measure with ordinal number 3 is now registered with a duration of 1/1 in part Part_POne (partID "P1", partName "Soprano"), fPartMeasuresWholeNotessVector.size () = 2
        The measure with ordinal number 4 is now registered with a duration of 1/1 in part Part_POne (partID "P1", partName "Soprano"), fPartMeasuresWholeNotessVector.size () = 2
%--------------------------------------------------------------
  Pass (ptional): displaying the LPSR as text
%--------------------------------------------------------------

... ... ...

    [PartGroup "PartGroup_1 ('0', fPartGroupName "Implicit")" (1 part), line 0
      fPartGroupName             : "Implicit"

Process 28676 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x1)
    frame #0: 0x00000001007b879a xml2ly`MusicFormats::msrPartGroup::print (this=0x0000600003e08100, os=0x000000010ca04650) const at msrPartGroups.cpp:1185:13
   1182       i      = iBegin;
   1183
   1184      for ( ; ; ) {
-> 1185        os << (*i);
   1186        if (++i == iEnd) break;
   1187        os << std::endl;
   1188      } // for
Target 0: (xml2ly) stopped.
(lldb)
```

This bug has thus been introduced between v0.9.60 and v0.9.61. Three have been several `git push` occurrences leading from v0.9.60 to v0.9.61.

70.1 Locating a bug at random in the Git log

git provides various ways to display the commits history of the repository through `git log` options, for example:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git log --pretty=format:"%h - %ad : %s"
ea338fd - Tue May 3 10:09:04 2022 +0200 : Complement to the Makefile
083db8a - Tue May 3 07:43:09 2022 +0200 : Before reverting to v0.9.60
12b6d93 - Mon May 2 09:41:37 2022 +0200 : Prior to bisecting
03d98be - Tue Apr 26 11:15:23 2022 +0200 : Finalized sone trace options
3dd7b72 - Tue Apr 26 10:10:38 2022 +0200 : Finalized sone trace options
7f7507c - Thu Apr 14 17:01:14 2022 +0200 : Finalized MFSL symtax and semantics, fixed a
    couple of issues
06109d3 - Thu Apr 14 17:00:50 2022 +0200 : Finalized MFSL symtax and semantics, fixed a
    couple of issues
62aa64c - Thu Apr 7 07:18:44 2022 +0200 : v0.9.62
671ffa4 - Thu Apr 7 06:26:34 2022 +0200 : Pre v0.9.62
bf9eb63 - Wed Apr 6 23:42:44 2022 +0200 : Pre v0.9.62
db4397c - Wed Apr 6 22:14:43 2022 +0200 : Pre v0.9.62
9a80b24 - Mon Apr 4 13:06:12 2022 +0200 : Added MFSL (MusicFormats Script Language)
2ef1150 - Mon Apr 4 12:07:07 2022 +0200 : Added MFSL (MusicFormats Script Language)
3f56d52 - Tue Mar 29 16:34:23 2022 +0200 : Added MFSL (MusicFormats Script Language)
fc1ea21 - Tue Mar 29 08:58:34 2022 +0200 : Added MFSL (MusicFormats Script Language)
737b996 - Mon Mar 28 23:42:03 2022 +0200 : Added MFSL (MusicFormats Script Language)
8c91155 - Sat Mar 26 08:35:55 2022 +0100 : Added MFSL (MusicFormats Script Language)
fc68a93 - Fri Mar 18 15:11:19 2022 +0100 : Added MFSL (MusicFormats Script Language)
01430a9 - Fri Mar 18 15:11:12 2022 +0100 : Added MFSL (MusicFormats Script Language)
4082813 - Thu Mar 17 18:50:02 2022 +0100 : Added MFSL (MusicFormats Script Language)
2696628 - Sun Mar 13 00:48:05 2022 +0100 : Added MFSL (MusicFormats Script Language)
a828231 - Thu Mar 10 14:28:11 2022 +0100 : Added MFSL (MusicFormats Script Language)
bf04937 - Wed Mar 9 12:53:17 2022 +0100 : Added MFSL (MusicFormats Script Language)
a855ee4 - Tue Mar 8 16:39:28 2022 +0100 : Added MFSL (MusicFormats Script Language)
b636816 - Mon Mar 7 14:49:54 2022 +0100 : Added MFSL (MusicFormats Script Language)
ecd5eaa - Sun Mar 6 00:11:05 2022 +0100 : Added 'keep-msr-voice, kmv' option
ec1c8ef - Sat Mar 5 08:48:39 2022 +0100 : Added 'ignore-msr-voice, imv' option
8246467 - Thu Mar 3 16:11:37 2022 +0100 : v0.9.61
603e19c - Thu Mar 3 13:43:48 2022 +0100 : Switched from C++11 to C++17 for <filesystem>
77d3d29 - Thu Mar 3 07:56:00 2022 +0100 : Switched from C++11 to C++17 for <filesystem>
a880063 - Thu Mar 3 07:44:08 2022 +0100 : Switched from C++11 to C++17 for <filesystem>
38b584f - Wed Mar 2 12:44:22 2022 +0100 : Switched from C++11 to C++17 for <filesystem>
662454a - Tue Mar 1 17:14:47 2022 +0100 : Pre-v0.9.61
2cb4d5f - Mon Feb 28 11:53:04 2022 +0100 : Renamed some documentation folders and files
c7839a8 - Mon Feb 28 09:56:46 2022 +0100 : Renamed some documentation folders and files
0e85f99 - Mon Feb 28 09:06:16 2022 +0100 : Renames some documentation folders and files
c5a43d9 - Fri Feb 25 17:48:29 2022 +0100 : Finalized files/musicxmlfiles/Makefile
21e3898 - Thu Feb 24 21:54:28 2022 +0100 : Added '-replicate-msr-measure'  option
9738598 - Thu Feb 24 21:53:00 2022 +0100 : Added '-replicate-msr-measure'  option
ae751c3 - Mon Feb 21 09:58:35 2022 +0100 : Added various options
f2d2f57 - Sat Feb 19 08:09:02 2022 +0100 : Workflow to publish Mac OS release
ac5ad6b - Sat Feb 19 08:00:57 2022 +0100 : Initializa npm package
29de34d - Fri Feb 18 11:00:42 2022 +0100 : v0.9.60
7c067d6 - Fri Feb 18 10:56:17 2022 +0100 : v0.9.60
5e3ba90 - Fri Feb 18 09:57:50 2022 +0100 : Pre v0.9.60
dfeb7be - Fri Feb 18 09:56:07 2022 +0100 : Pre v0.9.60
c31dde3 - Wed Feb 16 11:50:42 2022 +0100 : Updates to the make and cmake configuration
fd6fef0 - Wed Feb 16 09:45:44 2022 +0100 : Complements to the installation doc
b7ad2af - Tue Feb 15 17:40:53 2022 +0100 : Distrib test 17
50a904c - Tue Feb 15 17:37:53 2022 +0100 : Distrib test 16
cf65bd3 - Tue Feb 15 08:41:14 2022 +0100 : Distrib test 15
9cda15e - Tue Feb 15 08:38:57 2022 +0100 : Distrib test 14
74a2b7f - Tue Feb 15 08:30:21 2022 +0100 : Distrib test 13
```

```
ee011e9 - Mon Feb 14 15:06:59 2022 +0100 : Distrib test 13
4d6f9cb - Mon Feb 14 08:54:23 2022 +0100 : Distrib test 12
7692b7b - Mon Feb 14 08:52:50 2022 +0100 : Distrib test 12
d9e943d - Sat Feb 12 09:30:45 2022 +0100 : Distrib test 11
3dde810 - Sat Feb 12 09:29:51 2022 +0100 : Distrib test 11
f07b02a - Fri Feb 11 17:55:34 2022 +0100 : Distrib test 10
5113824 - Fri Feb 11 16:47:17 2022 +0100 : Distrib test 9
7f0fa8e - Fri Feb 11 16:45:21 2022 +0100 : Distrib test 8
93f72f4 - Fri Feb 11 16:32:33 2022 +0100 : Distrib test 6
121aa64 - Fri Feb 11 14:35:38 2022 +0100 : Distrib test 5
547556f - Fri Feb 11 11:30:17 2022 +0100 : Creation of MusicFormats repository
```

One can pick one of the commits, revert to it and check wether the bug is present in it.

70.2 Locating a bug in the commits with Git's bisection

Locating the particular push that introduced the bug can be facilitated by git's *bisect* facility. Here is how it works:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git bisect start

jacquesmenu@macmini: ~/musicformats-git-dev > git bisect good v0.9.60

jacquesmenu@macmini: ~/musicformats-git-dev > git bisect bad v0.9.61
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[0e85f994ab00ea2dd94ddcb1895cbae5a32f072a] Renames some documentation folders and files

jacquesmenu@macmini: ~/musicformats-git-dev > git branch
* (no branch, bisect started on v0.9.61)
  master
  v0.9.60
  v0.9.61
  v0.9.62
```

The bisection proposes commit `0e85f994ab00ea2dd94ddcb1895cbae5a32f072a` as a middle point between v0.9.60 and v0.9.61. So let us check it out:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git checkout 0
    e85f994ab00ea2dd94ddcb1895cbae5a32f072a
HEAD is now at 0e85f99 Renames some documentation folders and files

jacquesmenu@macmini: ~/musicformats-git-dev > git branch
* (no branch, bisect started on v0.9.61)
  master
  v0.9.60
  v0.9.61
  v0.9.62
```

Then, building this intermediate development version leads to:

```
jacquesmenu@macmini: ~/musicformats-git-dev/build > make
make macos
cd libdir && cmake .. -G Xcode   -Wno-dev
-- VERSION: v0.9.61

-- Configuring version v0
.v9
.v61

... ... ...

```

```
/Users/jacquesmenu/musicformats-git-dev/src/components/mfcLibraryComponent.cpp:12:10:
    fatal error:
      '../../../MusicFormatsVersionNumber.h' file not found
#include "../../../MusicFormatsVersionNumber.h"
         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 error generated.

... ... ...

** BUILD FAILED **
```

Well, this dev version had been pushed to have new files and/or contents saved on the MusicFormats repository . . . and we should try other commits around it.

A first possibility is to use `git bisect skip`, that moves to :

```
jacquesmenu@macmini: ~/musicformats-git-dev > git bisect skip
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[c7839a87549660963a8b1ef0898d5cbcce8305aa] Renamed some documentation folders and files
```

Checking commit c7839a87549660963a8b1ef0898d5cbcce8305aa out, we get:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git checkout
    c7839a87549660963a8b1ef0898d5cbcce8305aa
HEAD is now at c7839a8 Renamed some documentation folders and files
```

Building that leads to the same error as above. Let us skip one again:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git bisect skip
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[a880063c134a7ba49b31f5fb52b47f682058f64a] Switched from C++11 to C++17 for <filesystem>
```

It turns out the commit `a880063c134a7ba49b31f5fb52b47f682058f64a` does not build either. Let us skip to the next commit:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git bisect skip
Bisecting: 7 revisions left to test after this (roughly 3 steps)
[2cb4d5f2133d34a19fea8f47d9ed2ccfb24d0fad] Renamed some documentation folders and files

jacquesmenu@macmini: ~/musicformats-git-dev > git checkout 2
    cb4d5f2133d34a19fea8f47d9ed2ccfb24d0fad
HEAD is now at 2cb4d5f Renamed some documentation folders and files
```

Here the code base builds alright, and bug does not show up, so we should continue skipping.

70.3 Locating the bug in the code base

The bug we're after is found to have been introduced at this point:

- commit `2cb4d5f`, does not exhibit the bug;
- the next one, commit `662454a`, does.

The changes brought by commit can be shown with:

```
jacquesmenu@macmini: ~/musicformats-git-dev > git log --patch -1 662454a > patch_662454a.
    txt

jacquesmenu@macmini: ~/musicformats-git-dev > ls -sal patch_662454a.txt
256 -rw-r--r--@ 1 jacquesmenu  staff  80782 May  4 10:59 patch_662454a.txt
```

The bug shows up with `files/musicxmlfiles/fullbarrests/MeasureRestWithoutBarLine.xml`, but not with others such as `files/musicxmlfiles/multistaff/SATBExample.xml`.

Analysing the patch description in `patch_662454a.txt`, we find that nothing in the differences between those two successive patches can explain the crash. The problem thus lies elsewhere...

Part XVII

Indexes