THE SCINE MOLASSEMBLER DEVELOPERS:

JAN-GRIMO SOBEZ AND MARKUS REIHER

USER MANUAL

SCINE MOLASSEMBLER 1.1.0

ETH ZÜRICH

Contents

Introduction

Molassembler is a software package that aims to facilitate conversions between Cartesian and graph representations of molecules. It provides the necessary functionality to represent a molecule as a graph, modify it in graph space, and generate new coordinates from graphs. It can capture the absolute configuration of multidentate and haptic inorganic molecules from positional data and generate non-superposable stereopermutations as output.

SCINE Molassembler consists of a C++ library and a Python module which binds parts of the available C++ functionality.

In this manual, we describe the installation of the software and how to integrate it into any derivative projects. A prospect on features in future releases and references for further reading are added at the end of this manual.

Obtaining the software

Molassembler is distributed as open source software in the framework of the SCINE Project[1]. It is licensed under the BSD-3 clause license. Visit our website[2] to obtain the software.

[1] sciné.ethz.ch

[2] sciné.ethz.ch/download/molassembler

System requirements

Molassembler has only modest requirements regarding the hardware performance, and will chew through small problems on a single thread without much trouble. However, if the studied systems are large, it is worth enabling parallelization in order to reduce wall clock times.

Installation

Python bindings on PyPI

If you want to experiment with just the Python bindings, those are available as a package from PyPI for Linux platforms.[3] If you are using Linux, then you can install MOLASSEMBLER with:

[3] Caveat: Your CPU architecture must be x86–64 or i686.

```
python3 -m pip upgrade --user pip
python3 -m pip install --user scine_molassembler
```

When installed this way, the Python bindings are not optimized to your particular CPU instruction set and may therefore be slower than if you had compiled them yourself.

Conan package

Conan[4] is a Python-based package distribution solution. It integrates well with C++ code built with CMake, greatly simplifying the installation of libraries with multiple dependencies. Conan is installed with `pip`:

[4] `conan.io`

```
python3 -m pip install conan
```

MOLASSEMBLER packages are distributed from our research group's remote. You can add this remote with:

```
conan remote add scine https://scine-artifactory.ethz.ch/artifactory/api/conan/public
```

Before we install MOLASSEMBLER, consider the options the package provides. They are summarized below, with valid values and the default option value in bold face:

- `shared=[`**`True`**`, False]`: Build molassembler as a shared library
- `python=[True, `**`False`**`]`: Build the Python bindings
- `docs=[True, `**`False`**`]`: Build documentation

- `tests=[True, False]`: Build and run tests
- `microarch=[detect, none]`: Tune build to CPU instruction set

The base command to install MOLASSEMBLER is:

```
conan install scine_molassembler/1.1.0@
```

Changes to default options can be supplied right after `conan install` with `-o scine_molassembler:<option>=<value>`.

For instance, if you want Python bindings, your install command will be:

```
conan install -o scine_molassembler:python=True scine_molassembler/1.1.0@
```

Conan will proceed to identify your operating system, architecture and compiler. It will scan MOLASSEMBLER's dependencies and try to find a prebuilt package for your system. If any package is not available in binary form for your system, `conan` can handle its build with the addition of `-build=missing` as advised by Conan's error message. If the package is available in binary form, it will be downloaded.

From source code with CMake

In order to compile MOLASSEMBLER from source, you will need:

- a C++ compiler supporting the C++14 standard[5]
- `cmake` $\geq$ 3.9.0
- the `Boost` libraries $\geq$ 1.65.0 i
- the `Eigen` library $\geq$ 3.3.2, $<$ 3.4.0

[5] `gcc` $\geq$ 7.3.0, `clang` $\geq$ 4, `MinGW-w64`, etc.

Check your package manager for the relevant packages. For Ubuntu, for instance, you will need:

```
build-essential
cmake
libboost-all-dev
libeigen3-dev
```

First, we must get a complete set of source files. You can get a complete set of files with:

```
git clone --recursive https://github.com/qcscine/molassembler.git
```

or by extracting a downloaded release archive and executing

```
git clone https://github.com/qcscine/development-utils.git
```

within the `molassembler` folder.

Next, we'll get to configuring the source build:

```
mkdir build install
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=../install ..
```

This will look through the dependencies to make sure everything needed is there and prepare for building. We have specified an installation prefix on the command line that we will install into, which you can replace at your leisure.

If you additionally wish to compile and install the Python bindings, you can rerun `cmake` with the following flags added:

```
cmake -DSCINE_BUILD_PYTHON_BINDINGS=ON -DPYTHON_EXECUTABLE=$(which python3) ..
```

For further compile-time options regarding the build of Molassembler, you can run `cmake -L ..` in the `build` directory. Look for options prefixed with `SCINE` or `MOLASSEMBLER`. You can then build, test and install with:

```
make
make test
make install
```

In case you need support with the setup of Molassembler, please contact us by writing to scine@phys.chem.ethz.ch.

Note that Molassembler depends on further libraries that are distributed along or downloaded at configure time:

- RingDecomposerLib [6]
- nauty [7]
- JSON For Modern C++ [8]
- SCINE Utilities [9]

[6] Flachsenberg, F.; Andresen, N.; Rarey, M. RingDecomposerLib: An Open-Source Implementation of Unique Ring Families and Other Cycle Bases, *J. Chem. Inf. Model.* **2017,** *57,* 122–126; and Kolodzik, A.; Urbaczek, S.; Rarey, M. Unique ring families: A chemically meaningful description of molecular ring topologies, *J. Chem. Inf. Model.* **2012,** *52,* 2013–2021

[7] McKay, B. D.; Piperno, A. Practical graph isomorphism, II, *J. Symb. Comput.* **2014,** *60,* 94–112

[8] Lohmann, N. "JSON for Modern C++", 2013

[9] Bosia, F.; Brunken, C.; Grimmel, S. A.; Haag, M. P.; Heuer, M. A.; Simm, G. N.; Sobez, J.-G.; Steiner, M.; Unsleber, J. P.; Vaucher, A. C.; Weymuth, T.; Reiher, M. "qcscine/utilities: Release 2.0.0", 2020

Using the C++ library

Molassembler provides a Doxygen technical documentation[10] which documents the available functions in detail and also provides an overview of the available functionality. Here, we will only go into detail in how to link Molassembler into a new project with CMake.

[10] scine.ethz.ch/download/molassembler

In the CMake file of a new project, add the following:

```
# Suppose your target is named transmogrifier
find_package(molassembler REQUIRED)
target_link_libraries(transmogrifier PUBLIC molassembler::molassembler)
```

Note that if transmogrifier were a library you were making, PUBLIC here denotes that you use Molassembler types in your public interface, and any consumers of transmogrifier must also link against Molassembler, so alter this as needed.

Provided you have installed Molassembler somewhere, this should take care of include directories, compile definitions and linking for you. Perhaps you need to specify the path you have installed it to when invoking CMake using either of the arguments CMAKE_PREFIX_PATH or molassembler_DIR.

Using the Python module

MOLASSEMBLER provides Python bindings such that the core functionality of MOLASSEMBLER can be accessed also via the Python programming language. In order to build the Python bindings while compiling from source, you needed to specify `-DSCINE_BUILD_PYTHON_BINDINGS=ON` when running `cmake` during the installation (see also chapter Installation).

Depending on your method of installation, you may need to specify the path to the Python module in the environment variable `PYTHONPATH`, *e.g.*, you might need to run the command

```
export PYTHONPATH=$PYTHONPATH:<install directory>/lib64/python<version>/site-packages
```

where `<version>` is the Python version you are using (*e.g.*, 3.6) and `<install directory>` is where you installed MOLASSEMBLER. Now you can simply import the module and use it just like any other Python module. All functionality available in the Python module is documented with examples[11]. You can interactively explore the available functionality in an interactive Python shell using the `help` and `dir` Python functions:

[11] `scine.ethz.ch/download/molassembler`

```
import scine_molassembler as masm
dir(masm)
help(masm)
```

Extensions planned in future releases

- Better handling of aromatics
- Better ranking algorithm
- Stabilized SMILES parser

How to cite the software

We kindly ask you to cite the following reference in any publication of results obtained with MOLASSEMBLER.

J.-G. Sobez, M. Reiher. Molassembler: Molecular graph construction, modification and conformer generation for inorganic and organic molecules, *J. Chem. Inf. Model.*, 60, 3884 (2020) DOI: 10.1021/acs.jcim.0c00503.

Bibliography

[1] Flachsenberg, F.; Andresen, N.; Rarey, M. RingDecomposerLib: An Open-Source Implementation of Unique Ring Families and Other Cycle Bases, *J. Chem. Inf. Model.* **2017,** *57,* 122–126.

[2] Kolodzik, A.; Urbaczek, S.; Rarey, M. Unique ring families: A chemically meaningful description of molecular ring topologies, *J. Chem. Inf. Model.* **2012,** *52,* 2013–2021.

[3] McKay, B. D.; Piperno, A. Practical graph isomorphism, II, *J. Symb. Comput.* **2014,** *60,* 94–112.

[4] Lohmann, N. "JSON for Modern C++", 2013.

[5] Bosia, F.; Brunken, C.; Grimmel, S. A.; Haag, M. P.; Heuer, M. A.; Simm, G. N.; Sobez, J.-G.; Steiner, M.; Unsleber, J. P.; Vaucher, A. C.; Weymuth, T.; Reiher, M. "qcscine/utilities: Release 2.0.0", 2020.