Administrator Manual

for Macleod: A CLIF Parser, designed for Torsten Hahmann and Jake Emerson

R.C. Development

Designed by Reading Club Development:
Matthew Brown, Gunnar Eastman, Jesiah Harris, Eli Story

Version 1.1
April 28, 2023

R.C. DEVELOPMENT

Administrator Manual:

Table of Contents

1. Introduction

The Common Logic Interchange Format (CLIF) Parser is a capstone project for Dr. Torsten Hahmann and Jake Emerson, in partial fulfillment of the Computer Science BS degree for the University of Maine, completed by the Reading Club Development (RCD) team. Dr. Hahmann is a professor at the University of Maine with affiliation to the Spatial Data Science Institute and research interests in knowledge representation, logic, and automated reasoning. Jake Emerson works for Jackson Laboratories in Bar Harbor, Maine, where he would implement this parser into the workflow of projects he is involved with. RCD consists of four seniors from the University of Maine: Matthew Brown, Gunnar Eastman, Jesiah Harris, and Elijah Story.

RCD is responsible for the following products: MacleodCore, pre-existing code that will be used in the terminal; Macleod, code that has been adjusted by RCD that enables the scripts available in MacleodCore to be used as functions in a python library; and MacleodIDE, a plugin for the SpyderIDE that will allow the Macleod functions to have access to a User Interface, for an easy to understand experience.

The intended audience of this document consists of those who would, in the future, be making updates to the Macleod software, either to expand its potential even further or to update it in accordance with changes to the CLIF standard, or to changes to the file formats that Macleod can translate to.

In the second section of this document, we review relevant background information and system requirements for each edition of Macleod. In the third section we move into reviewing both how to install and operate Macleod, as well as the various administrative procedures for keeping Macleod up to date.

1.1 Purpose of This Document

The purpose of this Administrator Manual is to elaborate upon how one can install and manage the Macleod system.

1.2 References

[1] Brown, M, et al., CLIF Parser System Requirement Specification, 2022, https://github.com/Reading-Club-Development/macleod/blob/master/UMaine%20Capstone%20Documents/RCD_SRS_Ver_1.3.pdf.

[2] Brown, M, et al., CLIF Parser System Design Document, 2022,

https://github.com/Reading-Club-Development/macleod/blob/master/UMaine%20Capstone%20Documents/RCD_SDD_Ver_1.1.pdf.

[3] Brown, M, et al., CLIF Parser User Interface Design Document, 2022, https://github.com/Reading-Club-Development/macleod/blob/master/UMaine%20Capstone%20Documents/RCD_UIDD.pdf.

[4] Brown, M, et al., CLIF Parser Critical Design Review Document, 2022, https://github.com/Reading-Club-Development/macleod/blob/master/UMaine%20Capstone%20Documents/RCD_CDRD.pdf.

[5] Brown, M, et al., CLIF Parser Code Inspection Report, 2023, https://github.com/Reading-Club-Development/macleod/blob/master/UMaine%20Capstone%20Documents/RCD_CIR_Ver_1.1.pdf.

[6] Colore, Semantic Technologies Laboratory, http://stl.mie.utoronto.ca/colore/.

[7] Hahmann, Torsten, *Macleod*, GitHub, 2022, https://github.com/thahmann/macleod.

[8] International Organization for Standardization, *Common Logic* (CL): *a framework for a family of logic-based languages*, 2018, https://www.iso.org/standard/39175.html

2 System Overview

This section details what systems the Macleod system is available on and what Macleod is, especially related to Administration.

2.1 Background

Macleod is entirely a client-side process, so no intensive operations are normally necessary on the part of administrators. The responsibilities of the Administrators are to handle GitHub in accordance with the updating standards relating to the logical conventions involved in the Macleod system. There should not need to be any daily operations on the administration side, nor is daily maintenance necessary.

2.2 Hardware and Software Requirements

In order to use the system, any operating system in the current ecosystem (2023) will work. MacleodIDE requires Spyder 5 and Python 3.7 or later in order to use. MacleodCore also requires Python 3.7 or later. The plugin requires Python version 3.8 or later. The Macleod system is built to be run from the command line interface, within Python code segments, or within Spyder.

The Macleod system is extremely small, less than 100 megabytes, and is stored client-side.

3. Administrative Procedures

For the Macleod system, the maintenance required is threefold. Firstly, the logic behind the parser and scripts within MacleodCore must be kept up to date with the CLIF standard, along with also ensuring the standards of TPTP, LADR, OWL, and LaTeX are also kept up to date. Secondly, to maintain Macleod, every update to MacleodCore should accompany an update to the Macleod library. The Macleod library must also be kept up to date with Python's package specifications, and the Python programming language. Lastly, MacleodIDE, the Spyder plug-in, must simply not fall into obsolescence as Spyder updates over time.

3.1 Installation

Below we detail how one can install the various releases of Macleod. We will first discuss MacleodCore, which provides commands to be used from the terminal. Secondly, we will overview the installation process of Macleod, which can be installed as a python package. Last is how to install MacleodIDE, a plug-in for the Spyder IDE.

3.1.1 Installing MacleodCore

The installation of MacleodCore follows the same format as is in the Macleod GitHub repository [7]. We did not modify this procedure, but using only MacleodCore without the IDE or library is not recommended by Reading Club Development. The procedure is copied in its entirety here:

It is recommended to create a virtual environment to work out of:

On Windows hosts with Python3 already installed and on the %PATH%

```
# Create a new virtual environment called "ve" (or whatever you want to
name it)
> python -m venv ve
# Activate the newly created virtual environment
> ve/scripts/activate.bat
```

On Linux hosts with Python3 installed, the two commands can be run in one line:

```
>python3 -m venv ve && . ve/bin/activate
```

Once the virtual environment has been created and activated clone this repository and install:

```
# Clone the repository using https or ssh
> git clone https://github.com/thahmann/macleod.git
# Go to the folder that contains the cloned repository,
# e.g. "%USERPROFILE%\GitHub\macleod\" by default on Windows hosts,
# or a location like "/home/git/macleod" on Linux hosts
> cd macleod

# Install macleod and all dependencies via pip
> pip install .

# Optionally you can install the GUI components as well (see more information
about the GUI alpha version below)
> pip install .[GUI]
```

As a next step, the configuration files need to be put in place:

On Windows hosts:

```
# Go to your home directory and create a subfolder `macleod'
> cd %USERPROFILE%
> mkdir macleod
> cd macleod
# copy the configuration files from the github folder to this new
folder.
> copy "%USERPROFILE%\GitHub\macleod\conf\macleod_win.conf .
> copy "%USERPROFILE%\GitHub\macleod\conf\logging.conf .
```

On Linux hosts:

```
# Go to your home directory and create a subfolder `macleod'
> cd ~
> mkdir macleod
> cd macleod
# copy the configuration files from the github folder to this new folder
> cp "/home/git/macleod/conf/macleod_linux.conf .
> cp "/home/git/macleod/conf/logging.conf .
```

In a final step, add the binaries of the theorem provers and model finders to be used for ontology verification and proving of lemmas to the "macleod" directory and edit the configuration file "maleod_win", "macleod_linux" or "macleod_mac" as necessary. In particular the commands for the utilized theorem provers and model finders must use the correct paths (if not on the PATH variable) and commands. Likewise, make sure that in *logging.conf* the args parameter of the section *[handler_fHandler]* uses the correct path for your host.

Note: The provers are not needed for translation to TPTP, LADR *or for extraction of* OWL *ontologies.*

3.1.2 Installing Macleod

The installation of Macleod is as easy as installing a python library. In order to install the Macleod library, one must simply open the command line and run:

```
> pip install -i https://test.pypi.org/simple/ macleod==0.2
```

If you run the following command:

```
> pip show macleod
```

And get the following result:

```
Name: macleod
Version: 0.2.7
Summary: A simple way to parse clif files and convert them into different ontology
file formats
Home-page:
Author:
Author-email: Gunnar Eastman <gunnar.eastman@gmail.com>
License:
Location: [insert long filepath here]
Requires:
Required-by:
```

…then the Macleod python package has been installed correctly.

3.1.3 Installing MacleodIDE

In order to install the MacleodIDE, there are four steps. Firstly, one must install Spyder. The easiest way to install Spyder is with the Anaconda Python distribution, which comes with everything you need to get started in an all-in-one package. Download Spyder from its webpage. For more information, visit Spyder's Installation Guide. Spyder is the home of Macleod-IDE.

Second, Spyder will likely need to be updated. In order to update Spyder using conda, one can run from the command line (or Anaconda prompt on Windows):

```
> conda update anaconda
> conda update spyder
```

If this results in an error or does not update Spyder to the latest version, try:

```
> conda install spyder=5
```

The third step is to install the plugin using pip. The macleod_ide plugin is available via test.pypi.org. The plugin can be installed using pip by running:

```
> pip install -i https://test.pypi.org/simple/ macleod-ide
```

Lastly, it is necessary to set up a development environment. In principle, we could use any Spyder installed within a conda environment according to the instructions given in the installation guide. However, it is recommended to create a new environment which only has Spyder with the minimum dependencies needed for your plugin.

This minimum dependency environment can be installed in the following way:

```
$ conda activate base
$ conda install -c conda-forge mamba
$ mamba create -n spyder-dev -c conda-forge python=3
$ mamba activate spyder-dev
$ mamba install spyder
```

Then, you can launch the virtual environment and run

```
$ spyder
```

...and Spyder should open, your terminal should read: "spyder launched" , then "macleod_ide initialized!"

The plugin is the notepad symbol located at the top right hand corner of the screen.

3.2 Routine Tasks

The administrator must also keep Macleod up to date to the logical standards they wish to use by following the changelog of Macleod. In the event that there is a change to the library or plugin, the user will need to update or install the latest version of MacleodCore. The routine tasks for the Macleod system lie in the user maintaining their own virtual environments and such. In many cases, the user is the administrator because the system does not interface with the Internet, and thus the user is responsible for their local maintenance.

3.2.1 Updating Pypi Version

One task related to the maintenance of Macleod is ensuring that any updates to the source code are pushed to Pypi, so that those installing macleod can receive these updates. The process followed, while also explained here, is very similar to the one provided by python: https://packaging.python.org/en/latest/tutorials/packaging-projects/

The macleod folder, either downloaded from the git repository or via pip, should, for convenience, be placed into a folder that will contain the supplementary files, which we will refer to as the package folder. This is not necessary, but recommended for the sake of organization. The package folder can be any directory that contains the macleod folder.

If downloaded from the git repository, the macleod folder should contain a folder named "dist," a file named "README," and a file named "pyproject.toml." Move these three items into the package folder.

Open pyproject.toml in any text editor, and update the version number. At time of writing, it is present on line 7 of the document, as version 0.2.8. Save and close the file.

Open the terminal and navigate to the package folder, then run the following command:

```
py -m build
```

This will create two files in the dist folder: A .whl file and a .tar.gz file. Ensure that the only files in the dist folder are those that match the newest version number. Then run the following command:

```
py -m twine upload dist/*
```

This will ask for your pypi username and password, then upload the files within the dist folder to PyPi, making them available for distribution via pip.

Note that you may need to run `pip install --upgrade build` and `pip install --upgrade twine` if those libraries have not already been installed on your machine, and that on Linux or MacOS, "py" will need to be replaced with "python3"

3.3 Periodic Administration

For every update to TPTP, LADR, LaTeX, or CLIF standards, the Macleod system will have to be updated to accommodate the shifts in standards. Currently, the system is designed for the 2018 version of CLIF. Updates to the target formats can be performed by editing the to_tptp(), to_ladr(), or to_latex() functions of each file within the /logical folder. Should the OWL standards change, those changes can be reflected in dl/translation.py.

In the event that the CLIF standards change to make Macleod out of date, changes to update it must be made to the parsing rules in the /parsing/parser.py file.

3.4 User Support

User support can be achieved through either reading the extensive documentation accompanying this project, or by reaching out to Dr. Torsten Hahmann or Jake Emerson directly via emails found within the Macleod GitHub. At time of writing, Dr. Torsten Hahmann can be reached at torsten.hahmann@maine.edu and Jake Emerson at raymond.emerson@maine.edu. If an administrator would need contact with an RCD team member, please contact Matthew Brown at matthewbrownie3@gmail.com with any questions or concerns.

4. Troubleshooting

In this section, known bugs and limitations will be discussed, along with how to deal with potential errors or system failures.

4.1 Dealing with Error Messages and Failures

Error messages should be handled through the verbose error system already included with Python. If these errors cannot be resolved, the best course of action is to either reach out to Dr. Torsten Hahmann, for script errors or errors in logic, or Matthew Brown, for installation process errors.

4.2 Known Bugs and Limitations

The most evident limitation of the system is many places where code was never fully implemented, resulting in many functions that will only raise “not implemented” exceptions. These were present before RCD’s involvement, but are only limited to the translations to function-free prenex conjunctive normal form (ffpcnf) or the Web Ontology Language (OWL).

Appendix A

This appendix details the expectations that RCD shall uphold to the client upon completion of this document, and how future changes to this document shall be made.

RCD and the client, upon the signing of the document, are agreeing that this AM contains instructions on how an administrator may maintain the system. The client, in signing this AM, agrees that this document contains how to install and manage Macleod thoroughly and completely. The team, RCD, agrees that this is an extensive and complete record of how to manage and install Macleod.

Any changes made to this document must be approved by all members of RCD and the client via signatures to an additional appendix wherein the changes are enumerated and detailed. The signing of this appendix consents all members of RCD and the client that this structure of implementing changes is acceptable.

Name:	Signature:	Date:
Torsten Hahmann	________________	__/__/__
Jake Emerson	________________	__/__/__
Matthew Brown	________________	__/__/__
Gunnar Eastman	________________	__/__/__
Jesiah Harris	________________	__/__/__
Eli Story	________________	__/__/__

Appendix B

This appendix will contain the agreement that all members of RCD have read and consent to the document in its entirety.

Through signing this appendix, we, as the members of RCD, agree that we have reviewed this document fully, we agree to the formatting of this document, and agree to the content that is located within this AM. Each member of RCD may have minor disagreements with certain parts of this document, though by signing below, we agree that there are not any major points of contention within this AM. We have all agreed to these terms and placed our signatures below.

Name:	Signature:	Date:
Matthew Brown Comments:	________________	__/__/__
Gunnar Eastman Comments:	________________	__/__/__
Jesiah Harris Comments:	________________	__/__/__
Eli Story Comments:	________________	__/__/__

Appendix C

This appendix will outline the approximate contributions of each of the team members of RCD to the completion of this AM.

Matthew Brown

- Formatted the document
- Applied Sister team feedback
- Wrote the majority of the document

Gunnar Eastman

- Applied sister team feedback
- Wrote a significant amount of the document

Jesiah Harris

- Wrote some of the document

Eli Story

- Wrote some of the document
- Wrote sister team feedback