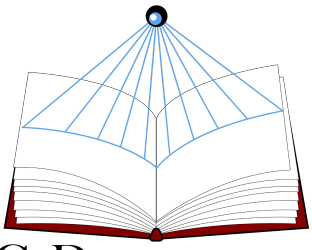
User Guide

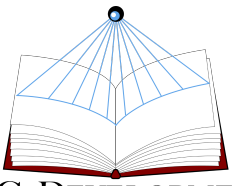
for Macleod: A CLIF Parser, designed for Torsten Hahmann and Jake Emerson



R.C. DEVELOPMENT

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

Version 1.0
Apr 18, 2023



R.C. DEVELOPMENT

User Guide:

Table of Contents

How to Use this Document

This user guide is constructed as follows:

Section 1 is an introduction to Macleod, its use cases, and its intended users. The introduction also contains information regarding the applicability and purposes of the user guide.

Section 2 provides a general overview of the functionalities available in Macleod and is intended to give readers/users a broad set of expectations on what will be encountered during the use of Macleod.

Section 3 consists of instructions for the use of Macleod and Macleod-IDE. This section is aimed at giving users an understanding of how to operate the system.

Section 4 is a list of references and documentation relevant to the functionalities and capabilities of Macleod.

Section 5 gives a list of common error messages that may be encountered during the use of Macleod, as well as troubleshooting methods and recovery procedures.

Section 6 is a glossary of terms related to Macleod and its intended use cases.

Related Documents

Num	Title	Author	Date	Issue
1	SRS - System Requirements Specification	R.C. Development	10/21/2022	1.3
2	SDD - System Design Document	R.C. Development	11/11/2022	1.1
3	UIDD - User Interface Design Document	R.C. Development	11/30/2022	1.0
4	CIR - Code Inspection Report	R.C. Development	3/10/2023	1.1
5	AM- Administrator Manual	R.C. Development	4/11/2023	1.1

1. Introduction

The CLIF (Common Logic Interchange Format) Parser is a stepping stone in the progress toward a unified Common Logic, which should allow for more properly defined variables, and hopefully slightly mitigating this crisis of reproducibility. Macleod stands for “Macleod Common Logic Environment for Ontology Development.” On a smaller scale, the purpose of Macleod is primarily to parse CLIF files to ensure they are syntactically correct according to CLIF standards, and translate them into other logic-based conventions. In addition to the CLIF Parser, Macleod has access to a GUI, Macleod IDE, in the form of a plugin for the Spyder IDE.

Macleod’s intended audience is specialists in scientific fields, and make use of TPTP, OWL, LADR, LaTeX, or CLIF files, and/or would be able to make use of an Ontology.

Use of the Macleod system assumes that prospective users are familiar to some degree with the use of one of the following: the command line, the Python programming language, or the Spyder IDE.

Applicability Statement

This User Guide is written for use with Macleod version 1.0 and its GUI, Macleod-ide version 0.0.2. Macleod 0.2.8 and its release history are available for viewing and installation here. Macleod-IDE and its release history are available for viewing and installation here.

Purpose

This user guide provides steps on how to use Macleod, a breakdown of related and relevant documentation, as well as a troubleshooting guide for issues commonly encountered when using Macleod. This user guide is intended for end users, casual users, and anyone else who wants to use Macleod or extend its features through further development.

Problem Reporting Instructions

To report any problems encountered during the installation, use, or maintenance of either Macleod or Macleod-IDE, please create an issue on the corresponding GitHub repository. If the problem encountered is urgent or important you may reach 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.

2. Overview

Below we describe the current available editions of the Macleod software. Use of MacleodCore assumes a familiarity with terminal commands. Use of Macleod assumes a familiarity with Python. Use of Macleod-IDE assumes a familiarity with the Spyder IDE.

Macleod

Macleod is a common logic parser that takes in CLIF files and creates axioms that are then used to create a file in a different logic-based convention. In essence, it translates one logic language to another.

The user will primarily use the one function `parse_clif()`. This function takes the path to a .clif file and the file type you would like to convert it to as a string. This will take in the file and create the axioms for that file. It will then take the axioms and send it off to the function corresponding to the desired file type to be converted to that file. This will then print out the location of the newly created file.

Macleod-IDE

Macleod-IDE is a Spyder plugin that utilizes the Macleod Pip library to parse files from the GUI of Spyder. The plugin creates a set of buttons that allows the user to select a file and click a button that will translate the file to the desired logical language. When the file is translated, the path to the newly created file will be printed in the terminal located in Spyder.

MacleodCore

MacleodCore is a series of python scripts that are designed to be run from the command line. These scripts have the same functionality as the Macleod python package functions.

3. Instructions

Below are detailed instructions for the installation and use of Macleod the python package, MacleodCore the collection of command-line scripts, and MacleodIDE the Spyder plugin.

Macleod Installation

Because Macleod is a Pip library, installation is very simple. All the user needs to do is run the following command in the terminal:

```
pip install macleod
```

This will install Macleod for use in any file.

Macleod Use

To use Macleod once it is installed, the user can import Macleod into any file using the following import statement:

```
import macleod.scripts.parserlib as macleod
```

In this import statement, the macleod after “as” can be anything the user would like. This is simply the variable that will be used to invoke Macleod functions.

To use Macleod once imported, the user can use the following function call:

```
macleod.parse_clif(“FILEPATH”, “LANGUAGE”)
```

As in the import statement, the macleod is the name the user decided on. Make sure that is the same as the variable name after the “as” in the import statement.

The function `parse_clif()` takes two arguments. The first is the file path, in string form, to the .clif file the user wishes to translate. Please use the full filepath where possible. The second argument is the language the user wishes the output file to be translated to. It also is a string. The following are the languages available at this current time:

- “TPTP”
- “LADR”
- “LaTeX”

The Language argument is optional, and if no argument is provided, the .clif file will be parsed and printed, without being translated.

Macleod-IDE Installation

Macleod-IDE requires Macleod (installation instructions above) and the Spyder IDE to be installed.

To 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 it from its webpage. For more information, visit its Installation Guide.

Update Spyder using conda:
From the command line (or Anaconda prompt on Windows), run:

```
$ 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
```

Install the plugin using pip:
The macleod_ide plugin is available via pypi.org

The plugin can be installed using pip as follows:

```
$ pip install macleod-ide
```

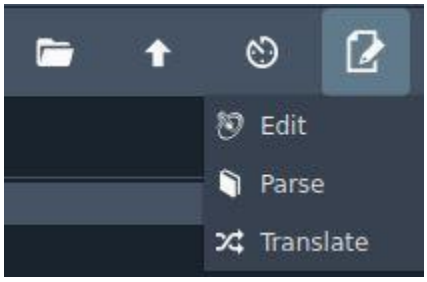
Set up a development environment:
In principle, we could use any Spyder installed within a conda environment according to the instructions given in the installation guide. However, it is recommended to create a new environment which only has Spyder with the minimum dependencies needed for your plugin.

It can be installed in the following way:

```
$ conda activate base
$ conda install -c conda-forge mamba (Recommended by spyder-ide.org)
$ mamba create -n spyder-dev -c conda-forge python=3
$ mamba activate spyder-dev
$ mamba install spyder
```

To use Macleod-IDE:

After installation, Macleod-IDE will be available for use in Spyder. The plugin icon will appear in the top right corner as shown here:



Macleod-IDE offers the following functionalities, though they are nonfunctional at this point in development:

Edit: Creates pop-out window for the loaded CLIF file, highlights errors, and allows for editing of the CLIF file.

Parse: Parses the loaded CLIF file and ensures that it is syntactically correct.

Translate: Parses the loaded CLIF file, ensures that it is syntactically correct, and translates the CLIF file to an indicated other language.

MacleodCore Installation

The installation of MacleodCore has two potential methods, the first follows the same format as is in the Macleod GitHub repository, and involves downloading the code in that repository and installing it. The second involves downloading Macleod via pip, and using that library similarly to the code in the GitHub repository. We did not modify this first 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:

```
> python -m venv ve
> ve/scripts/activate.bat
```

On Linux hosts with Python3 installed:

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

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

```
> git clone https://github.com/thahmann/macleod.git
> cd macleod
> pip install .
```

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.

The second method of installing MacleodCore is to install Macleod via pip, then navigate in the terminal to the location of that python library in the terminal, then run

```
> pip install .
```

The library is likely present under C:\Users\[username]\AppData\Local\Packages\PythonSoftwareFoundation.Python.[long version number]\LocalCache\local-packages\Python[short version number]\site-packages\macleod

Reference Section

In this section are explanations of each terminal command for MacleodCore, as well as each function made available via Macleod.

Macleod

In the Macleod library, there are two functions designed for external use. They will be listed below alongside all of their arguments. ‘(req.)’ denotes a required argument, and ‘(opt.)’ denotes an optional argument.

```
parse_clif(filepath, format="None", OWL_version="Full", enum=False, output=False, resolve=False, nocond=False, base=None, sub=None):
```

parse_clif() is the first of these functions, and will accept the following arguments. Note that the file path is the only required argument.

```
#Returns nothing
#Filepath (req.): A string filepath of the file or folder to be converted
#format (opt.): The format to be converted into. Not case sensitive. If set to
"None" it'll just print the clif file.
#enum (opt.): Do you want it to enumerate axioms? Only relevant for LaTeX output.
#output (opt.): Boolean. Do you want it to write an output file (True) or just
print (False)
#resolve (opt.): do you want it to resolve import statements
#nocond (opt.): do u want it to keep conditionals (True) or switch to
disjunctions (False)
#base (opt.): path to directory with ontology files. Only relevant if resolve is
on
#sub (opt.): path to directory with import files. Only relevant if resolve is on
```

Note that 'format' should be given one of the following values, if it is given a value at all:

- "TPTP"
- "LADR"
- "LaTeX"

```
check_consistency(filepath, method="simple", output=True, resolve=False,
stats=True, nontrivial=False, base=None, sub=None):
```

check_consistency() is the second function provided by Macleod, and exists to help confirm that a given CLIF file is not only syntactically correct, but also logically consistent.

```
#Function to check the consistency of ontologies in the Common Logic Interchange
Format (.clif).'
    #filepath: path to the file or folder of files to be checked.
    #method: string determining how to check
        #"simple" for a simple consistency check
        #"full" for a full recursive consistency check in case the entire
ontology is not probably consistent
        #"module" to check each module individually
    #output: Bool. Do you want it to write to an output file
```

```
    #resolve: Bool. auto resolve imports?
    #stats: bool. Do you want detailed stats (including definitions) about the
ontology?
    #nontrivial: bool. Instantiate all predicates to check for nontrivial
consistency?
    #base: Path to directory containing ontology files (basepath; only relevant
when option --resolve is turned on; can also be set in configuration file
    #sub: String to replace with basepath found in imports, only relevant when
option --resolve is turned on
```

MacleodCore

Once installed, MacleodCore offers scripts that fill similar roles to the two Macleod functions.

On the command line, parse_clif should be constructed thusly:

```
parse_clif -f [FILEPATH] (--[LANG]) (--enum) (-out) (--resolve) (--nocond)
(--base) (--sub) (--ffpcnf) (--clip)
```

-f and the accompanying filepath is again the only required argument. Though an optional argument, if a value is provided, the only accepted inputs for LANG are as follows:

- --owl
- --tptp
- --ladr
- --latex

Below, explanations for individual optional arguments are provided.

```
'--enum', 'Enumerate axioms in LaTeX output'

'--output', 'Write output to file'

'--resolve', 'Automatically resolve imports'

'--nocond', 'Do not use conditionals (only applies to TPTP, LADR and LaTeX
production)'
```

```
'--base', 'Path to directory containing ontology files (basepath; only relevant
when option --resolve is turned on; can also be set in configuration file)'

'--sub', 'String to replace with basepath found in imports, only relevant when
option --resolve is turned on'

'--ffpcnf', 'Automatically convert axioms to function-free prenex conjunctive
normal form (FF-PCNF)'

'--clip', 'Split FF-PCNF axioms across the top level quantifier'
```

The second script is check_consistency, which can be called like so:

```
check_consistency -f [FILEPATH] (--output) (--resolve) (--stats)
(--nontrivial) (--base) (--sub) --[METHOD]
```

Note that filepath and method are the only two required arguments. Below, each of the optional arguments are elaborated upon:

```
'--output', 'Write output to file'

'--resolve', 'Automatically resolve imports'

'--stats', 'Present detailed statistics (including definitions) about the
ontology'

'--nontrivial', 'Instantiate all predicates to check for nontrivial consistency'

'--base', 'Path to directory containing ontology files (basepath; only relevant
when option --resolve is turned on; can also be set in configuration file)'

'--sub', 'String to replace with basepath found in imports, only relevant when
option --resolve is turned on'

The following are the options for the ‘method’ argument, and are mutually
exclusive:

'--simple', 'Do a simple consistency check'
```

```
'--full', 'Do a full recursive consistency check in case the entire ontology is
not probably consistent'

'--module', 'Check each module individually'
```

Error Messages and Recovery Procedures

It should be the case that the only error messages Macleod produces are related to incorrect input files, in the case that those input files do not correctly implement the CLIF syntax. In the case that there is such an error, the system will note the line at which this error occurs, and will print the beginning of the file being parsed, and the filename.

```
2023-04-09 09:38:02,706 macleod.scripts.parserlib      INFO     Starting to parse macleod/src/macleod/tests/rcc_basi
clif
Error at line 25! in: /*******************************************************************************
 * Copyright (c) University of Toronto and others. All rights reserved.
 * The content of this file is licensed under the Creative Commons Attribution-
 * ShareAlike 4.0 Unported license. The legal text of this license can be
```

The recommended solution is to consult the CLIF Standard, provided by the ISO here: here.

Glossary

Axiom - A statement or proposition that is taken to be true to be used in further reasoning and arguments

CLIF - Common Logic Interchange Format. ISO/IEC 24707:2007 defines Common Logic: a first-order logic framework intended for information exchange and transmission. More information available here.

LADR - Library for Automatic Deduction Research, a file format.

Ontology - A collection of classifications and traits used to organize information.

OWL - Web Ontology Language, a file format.

Parse - A form of syntactic analysis. To divide a string into grammatical parts and check the correctness of its syntax based on the rules of a logical language's grammar.

TPTP - Thousand Problem Theorem Prover, a file format.

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 UG contains instructions on how an administrator may maintain the system. The client, in signing this UG, 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	_____	__/__/__
Elijah 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 UG. 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:	_____	__/__/__
Elijah Story Comments:	_____	__/__/__

Appendix C

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

Matthew Brown
- Formatted the document
- Did some editing of the document

Gunnar Eastman
- Handled the majority of the editing
- Wrote some of the document

Jesiah Harris
- Wrote a good portion of the document
- Did some sister team editing

Elijah Story
- Wrote a good portion of the document
- Conducted sister team review