Macleod: A Common Logic Environment for Ontology Development

— Installation Instructions —

Torsten Hahmann

August 17, 2014

1 The COLORE tools

The Macleod tools have been developed to assist with ontology design, exploration, and verification by offering a set of scripts and associated tools that read ontologies specified in Common Logic, aggregate imported ontologies, translate them to the TPTP format (now the de-facto standard input format for automated theorem provers specified by the problem library "Thousand Problems for Theorem Provers " accessible at `tptp.org`) and the LADR format (used mainly by the automated theorem prover Prover9-Mace4), check their consistency and nontrivial consistency, and prove desired properties. Consistency checking and proving on properties relies on using existing theorem provers (currently Vampire and Prover9) and model finders (currently Paradox and Mace4) in parallel, looking for both a positive ("found a proof" or "found a model") and negative answers ("found a counterexample", "proved inconsistency").

The toolset's functionality can be easily extended by adding new scripts that build on existing functionality (such as translation and aggregation of modules) but also by plugging in additional theorem provers and model finders in a configuration file. All scripts can be run using either Windows and Linux.

The purpose of this document is to provide step-by-step installation instructions and give a brief overview of the use of the configuration options and how to run scripts.

2 Setup

Follow these instructions to install Macleod on a Windows machine (tested with Windows 7 and 8). For installation on Linux, follow the same instructions but skip the Windows specific steps and replace all paths by appropriate Linux paths.

Step 1: Install Python 2.7 and Python for Windows You will need a local installation of Python 2.x (tested with 2.7.1 and 2.7.8; the scripts rely on libraries that will not work with Python 3.x). If you do not have Python 2 already installed, download and follow the instructions from `http://www.python.org/download/releases/2.7/`. For later reference, we assume that you install Python to the directory `C:\Python27`.

You will also need the "Python for Windows extension" (called `pywin32`) from `http://sourceforge.net/projects/pywin32/files/`. If your Python installation is 64-bit, choose the 64-bit version of pywin, otherwise use the 32-bit version of pywin.

Finally, you will need to install the "WMI module", a wrapper for the Python for Windows extension (pywin32), from `http://timgolden.me.uk/python/wmi/index.html`.

Step 2: Install Theorem Provers and Model Finders You will need to locally install the theorem provers and model finders you which to use. While some functionality of Macleod works without them, proving properties or consistency relies on them. So far, Macleod has successfully utilized Prover9-Mace4, Paradox, and Vampire. Due to licencing restrictions, the theorem provers and model finders cannot be distributed as part of Macleod and must be installed individually by the user.

For simplicity, we assume that the executables are placed in the common folder `C:\Reasoning`. This includes `paradox3.exe`, `vampire_win.exe`, and the `bin` folder from Prover9-Mace4, subsequently called `prover9-bin`.

You can download Prover9-Mace4, see `http://www.cs.unm.edu/~mccune/mace4/download/` and Vampire from `http://www.vprover.org/download.cgi`. The sources for Paradox are available from `http://www.cse.chalmers.se/~koen/code/`. If you need the Windows binary, contact the author of Macleod.

Step 3: Install Macleod Download the Macleod sources from `https://github.com/thahmann/macleod` and copy the folders `conf`, `pyparsing`, `src`, and `tasks` to the directory `C:\Reasoning\macleod`.

Depending on where you placed your Macleod installation and the theorem provers and model finders, you may need to edit the files

- `C:\Reasoning\macleod\conf\macleod_win.conf`
- `C:\Reasoning\macleod\conf\logging.conf`
- `C:\Reasoning\macleod\src\filemgt.py`

in the following way.

Edit the following entries in `C:\Reasoning\macleod\conf\macleod_win.conf`:

- `path`, `subprocess_log` in the section `system`;
- `prover9`, `mace4`, `vampire`, `paradox` in the section `command`;
- ensure that only the theorem provers and model finders that you installed locally are listed in the entries `provers` and `modelfinders` in the section `active`.

In `C:\Reasoning\macleod\conf\logging.conf`, you may need to edit the entries:

- `args` in the section `handler_fHandler`,
- `file` in the section `subprocess_logging`.

In `C:\Reasoning\macleod\src\filemgt.py`, you may need to edit the variable assignment `config_dir = 'C:\Reasoning \macleod \conf'` on line 15.

Step 4: Add directories to local PATH environment variable Add the directories where the Python executable resides (`C:\Python27`) to the `PATH` variable. In Windows, open `My Computer` (or `This PC`) `-> Properties -> Advanced System Settings -> Environment Variables` and add the path `C:\Python27` to the variable `Path` either in `User variables` or `System variables`. If the variable already exists, add the directory to the end, using a semicolon as separator, otherwise create a new variable with that name.

Use the command "echo %PATH%" in the command line to ensure your edits are correct.

Step 5: Add directories to local PYTHONPATH environment variables In the same way as in the previous step, edit or create a variable called `PYTHONPATH`. You need to include the paths `C:\Python27` and `C:\Reasoning\macleod` in `PYTHONPATH`.

Again, use the command "echo %PYTHONPATH%" in the command line to ensure your edits are correct.

Having the `PYTHONPATH` variable point to the folder of the Macleod installation will allow you to call scripts by their name, omitting the full path. For example, we can use (in Windows)

```
check\_consistency.py input\_file
```

or (in Linux)

```
python -m check\_consistency input\_file
```

instead of

```
python C:\Reasoning\macleod\scripts\check\_consistency. input\_file
```

Editing your environment variables in Linux If you are using the TC shell (`tcsh`) as your default shell in Linux, you can change the environment variables by the following to your configuration file `.tcshrc` located in your home directory. Make sure that you keep all other directories that were previously in the `PATH` variable.

```
setenv PATH /usr/bin:/usr/local/bin/:/torsten/macleod/
setenv PYTHONPATH /torsten/macleod
```

3 Configuration options

The following settings are configured in the main configuration file `C:\Reasoning\macleod\conf\macleod_win.conf`.

3.1 Input Common Logic ontologies and ontology modules

The section `cl` contains the following entries:

`prefix` the namespace that all ontologies are used and that is ignored when resolving imports in the ontology. For example, the default setting of `http://colore.oor.net` means when the ontology `http://colore.oor.net/multidim\_space\_codi/codi.clif` is processed, the imported module `http://colore.oor.net/multidim\_space\_codi/codi\_basic.clif` is expected to be located in the same directory as the importing file `codi.clif`, while the imported module `http://colore.oor.net/multidim\_space\_cont/definitions/c.clif)` is expected to be located in the directory `..\multidim_space_cont\multidim_space_cont` relative to `codi.clif`'s location.

`ending` the file name ending that all ontology files are expected to have (default: `.clif`)

`definitions_subfolder` the subfolder (relative to an ontology) that contains ontology modules whose only purpose is to define a new symbol. This information is only used to distinguish primitive from defined symbols in the ontologies.

`theorems_subfolder` the subfolder (relative to an ontology) where each ontology module contains one or multiple properties that we would like to prove about an ontology. Those modules most import the ontology that the belong to.

`interpretations_subfolder` the subfolder (relative to an ontology) where relative interpretations relationships to other ontologies are located **Currently not in use**

`consistency_subfolder` the subfolder (relative to an ontology) where logical extensions that test for nontrivial consistency will be placed.

`mappings_subfolder` the subfolder (relative to an ontology) where mappings of all its symbols in terms of another ontology's symbols are located **Currently not in use**.

3.2 Translations

For the translations to LADR and TPTP syntax, the sections `ladr` and `tptp` specify the folder where all translations are placed (relative to the original Common Logic file's location, `conversions` by default) and the ending that is automatically assigned (by default `.p9` and `.tptp`. The entry `all_ending` specifies the additional name that is added to a cumulative translation of a Common Logic ontology. Such a cumulative translation contains the axioms of all imported ontology modules as well. Setting this configuration to `.all` as by default results in the file `ontology.clif` to be translated to `conversions\ontology.all.tptp`.

3.3 Symbol Replacements

While Common Logic is very flexible in the use of non-alphanumeric names for relations, functions, and constants, TPTP is not. For this reason, all relation, function, and constants with non-alphanumeric names are substituted automatically with names of the form `clifsymX` where X is a numeric identifier. Often, this makes the translations much less accessible to humans. For this reason, one can specify more meaningful replacements in `src\clif.py` using the variable `SYMBOL_TRANSLATIONS` starting on line 55. The variables is a set of pairs, the first element indicated the symbol to be replaced and the second element its alphanumeric replacement symbol.

4 Tasks

4.1 Translating a Common Logic theory to Prover9 syntax

Use the script `clif_to_ladr.py` to translate an ontology to the LADR syntax. All imported modules are also translated, each module is located in a subdirectory (as specified by the entry `folder` in the `ladr` section of the configuration, the subdirectory is called "conversions" by default) from where the original Common Logic file is located.

The available options restrict the translation to a single ontology and its imports (`-single`) or to a single ontology module (`-module`). If all translations are to be cumulated into a single output file, use the option `-cumulate`.

If one wants to translate all ontologies within a folder (and, recursively, all its subfolders), the script `clif_to_ladr_all.py` can be used. All options (currently `-single`, `-module`, `-cumulate`) available for `clif_to_ladr.py` can be used.

4.2 Translating a Common Logic theory to TPTP syntax

Use the scripts `clif_to_tptp.py` and `clif_to_tptp_all.py` to translate an ontology or a set of ontologies to the TPTP syntax. The same options as for `clif_to_ladr.py` are available.

4.3 Proving a lemma or a set of lemmas

We use the `-l` option, e.g.:

```
python -m ColoreProver dim/lemmas/inc_p_lemmas -l
```

It is assumed that all ontologies necessary are imported by the file `dim/lemmas/inc_p_lemmas`, i.e., the axioms directly included in `dim/lemmas/inc_p_lemmas` are the sentences to be proved, while all imports (including the import closure) are the axioms we are allowed to use to produce a proof. If `dim/lemmas/inc_p_lemmas` contains more than a single sentence, they will be broken into individual files and each will be proved separately.

4.4 Testing a set of heuristics for proving a lemma

We use, e.g.:

```
python -m ColoreProver dim/lemmas/inc_p_lemmas -l -t
```

Each Common Logic module is assumed to have the filename `modulename.clif` with the namespace defining the directory. For example, the module `inc/axioms/lineparts_segs` should be located in the directory `inc/axioms` (relative to the current path) and named `lineparts_segs.clif`.