2024-08-07 Guidance of Frama-c

Frama-c

Installation

opam is the OCaml package manager. Every Frama-C release is made available via an opam package.

First you need to install opam, then you may install Frama-C using opam.

Installing opam

Several Linux distributions already include an `opam` package.

macOS has opam through Homebrew.

Windows users can install opam via WSL (Windows Subsystem for Linux).

If your system does not have an opam package, you can use the provided opam binaries available at:

http://opam.ocaml.org/doc/Install.html

Note: the `opam` binary itself is very small, but the initialization of an *opam switch* usually takes time and disk space, since it downloads and builds an OCaml compiler. Please refer to the opam documentation for details.

Installing Frama-C (including dependencies) via opam

The Frama-C package in opam is called `frama-c`. It includes:

- the command-line `frama-c` executable;
- the graphical interface, `frama-c-gui` (in supported systems);
- `ivette`, Frama-C's new Electron-based GUI.

Note: Ivette's dependencies are *not* included in the opam package, but downloaded from `npm` when the user runs `ivette` for the first time.

`frama-c` has some non-OCaml dependencies, such as GMP. opam includes a mechanism (`depext`) to handle such dependencies. It may require administrative

rights to install system packages (e.g. `libgmp`).

If your `opam` version is >= 2.1, such dependencies are installed automatically when installing `frama-c` itself:

```
# install Frama-C and dependencies with opam >= 2.1
opam install frama-c
```

If your opam version is < 2.1, you need to install `depext` first, then use it to install Frama-C's dependencies:

```
# install Frama-C's dependencies with pre-2.1 opam
opam install depext
opam depext frama-c
# then install Frama-C itself
opam install frama-c
```

If there are errors due to missing external dependencies, opam usually emits a message indicating which packages to install. If this is not sufficient, there may be missing dependencies in opam's packages for your system. In this case, you may create a Gitlab issue indicating your distribution and error message.

Test Installation

Test the frama-c installation by the following code to generate a call graph. Before that, you should prepare a C file: test.c.

```c
// test.c
#include <stdio.h>

void func(){
printf("hello Zihuatanejo\n");
}

int main(){
func();
return 0;
}
```

Then input the following command.

```shell
frama-c -cg test.dot test.c
```

If frama-c is installed successfully, you will find a test.dot file.

Plugin: PCGE

Github Link

Requirement: Dune

The recommended way to install dune is via the opam package manager:

```
$ opam install dune
```

You can also build it manually with:

```
$ make release
$ make install
```

Usage

Do not follow the readme in PCGE repo, it is out of the date.

Install Plugin

```shell
git clone https://github.com/sduprat/PCGE
```

Prepare Our Workspace

```shell
# in the same folder
mkdir workspace
cd workspace
cp ../plug.ml .
touch dune dune-project
```

The content of `dune` is as following:

```
(library
 (name Pcge)
 (public_name frama-c-pcge.core)
 (flags -open Frama_c_kernel :standard)
 (libraries frama-c.kernel))

(plugin
 (name pcge)
 (libraries frama-c-pcge.core)
 (site (frama-c plugins)))
```

The content of `dune-project` is as following:

```
(lang dune 3.16)
(using dune_site 0.1)
(name frama-c-pcge)
(package (name frama-c-pcge))
```

Remember that, you should change `lang dune 3.16` to your **dune version**

Plugin Compile

SHELL

```
# in workspace folder
dune build --profile release
```

Test the Plugin

SHELL

```
touch main.c func.c func.h
```

The content of `main.c` is as following:

C

```c
#include <stdio.h>
#include "func.h"

int main(){
    myFunction();
    return 0;
}

int test(){
    myFunction();
    return 0;
}
```

The content of `func.h` is as following:

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

void myFunction();

#endif
```

The content of `func.c` is as following:

```c
#include <stdio.h>
#include "func.h"

void myFunction(){
    printf("Hello Zihuatanejo\n");
}
```

Generate call graph with the following command:

```shell
dune exec -- frama-c -debug 3 ./test_code/main.c ./test_code/func.c -pcg-m m.dot -pcg-f f.dot
```

`m.dot` describes the connections among the `*.c` files; `f.dot` describes the connections among all of the functions in `*.c` files

In the end, good luck