Inspect: A Runtime Model Checker for Multithreaded C Programs

May 2, 2009

`Inspect` is a tool for systematic testing multithreaded C programs under specific inputs. It can systematically explore different interleavings of multithreaded programs under specific inputs by repeatedly executing the program. `Inspect` is aimed to reveal concurrency related bugs, such as data races, deadlocks, etc. (Because of the lack of a C++ source code transformer, Inspect cannot automatically instrument C++ programs. With manual instrumentation, we can still use Inspect to check multithreaded C++ programs. But you need to know more about how Inspect works to do the manual instrumentation.)

1 Download and Installation

The file inspect-{*version*}.tar.gz contains the complete source distribution. Inspect is written in C++ and Ocaml. You need Ocaml release 3.10 or higher, and a C++ compiler to build Inspect. Inpect has been tested under Linux systems. It has not been tested under Windows or other Unix systems yet. The source can be compiled with the following commands:

```
tar xzf inspect-{version}.tar.gz
cd inspect-{version}
make distclean
make
```

2 How to Use Inspect

The following command line shows how to use Inspect. First we instrument the multithreaded code. Then we compile the instrumented code, and link with a wrapper library. After this, we have an executable which can be systematically tested using inspect.

```
bin/instruemnt   examples/dpor-example1.c
bin/compile      dpor-example1.instr.c
./inspect ./target
```

2.1 Emacs Interface

The Emacs interface is found in the file `inspect-mode.el`. The file `emacs-config`, generated by `make`, contains a minimum emacs configuration whose contents should be added to the user's emacs startup script.

Customization variables:

- `inspect-path` is the absolute path the the Inspect directory.
- `inspect-instr-user-args` is a list of string arguments to pass to the instrumentation.
- `inspect-run-user-args` is a list of string arguments to pass to Inspect.

To run Inspect, select the a program source buffer and invoke the command (M-x) `inspect-run`. The buffer `*inspect-run*` will display progress information including the current test run and the number of races and deadlocks currently found. Inspect may be terminated at any time by pressing `k` in the `*inspect-run*` buffer.

The `*inspect-run*` buffer displays the traces corresponding to the first run (`*inspect-first-run*`) and all races (`*inspect-`*run*`-race-`*count*`*`), deadlocks (`*inspect-`*run*`-deadlock*`), and assertion failures (`*inspect-`*run*`-assert*`). It also includes the output of the program (`*inspect-`*run*`-output*`) if there is any.

Pressing enter or clicking the left mouse button will display the selected buffer. Pressing backspace will return to the main buffer `*inspect-run*`. Alternatively, buffers may be selected using `C-x B` or `C-x C-b`. The source line of a trace entry can be displayed by pressing enter or clicking the left mouse button.

Debugging buffers:

- `*inspect-compile-debug*` contains the command line for the instrumentation and its output.
- `*inspect-run-debug*` contains the command line for the invocation of Inspect and the uninterpreted output.

3 Related Projects

- Verisoft
- CHESS