Ceagle+ VM Manual

For CAV 2017 Review

About

- This manual is used to review:
  - Ceagle+: A Structural Abstraction-Based C Program Verifier
- Ceagle+ website:
  - http://sts.thss.tsinghua.edu.cn/tool/
- Ceagle+ demonstration video for CAV 2017:
  - https://youtu.be/4Jk8eRCuRmo
- Miscellaneous
  - *benchexec* is used in this VM to measure accurate time & memory usage
    - https://github.com/sosy-lab/benchexec

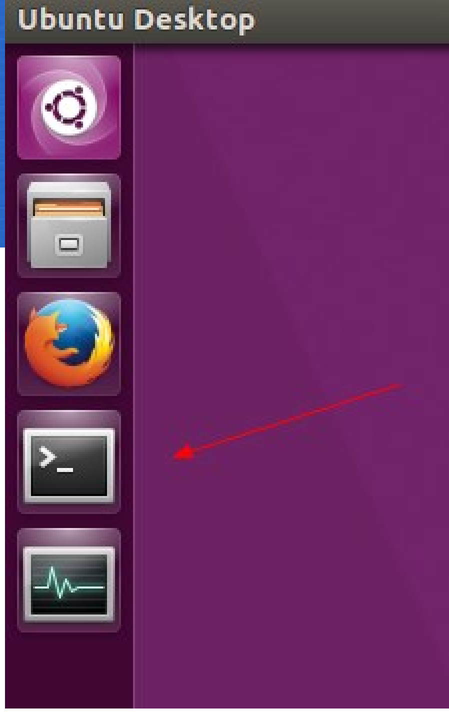
1. Virtual Machine Image

- Image link:
- Virtual machine name:
    - Ceagle-Plus-VM
        - this name will be used afterwards to stand for the virtual machine
- Username:
    - user
- Password:
    - user

2. Where can I find Ceagle+?

- Inside Ceagle-Plus-VM:
  - /home/user/Downloads/ceagle-plus/
    - including files:
      - LICENSE (text file)
      - sv-ceagle (binary)
      - z3 (binary)
- Download from website:
  - http://sts.thss.tsinghua.edu.cn/tool/data/cav-2017/ceagle-plus.tar.gz

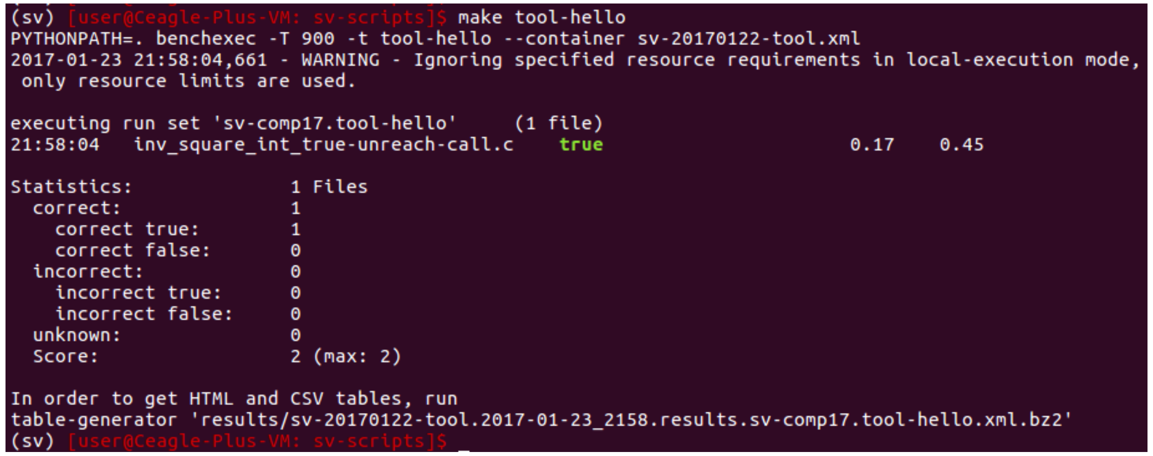
3. Setup



1. Open a Terminal app in Ceage-Plus-VM desktop GUI
2. Just type commands below **as-is** in the terminal:

```
$ cd ~/Documents/sv-scripts/
$ source virtualenvwrapper.sh
$ workon sv
(sv) $ ./fix.sh
(sv) $ export PATH=$PATH:~/Downloads/ceagle-plus/
(sv) $ make tool-hello
```

3. If you see some thing like this, then Ceagle+ runs well on your side.



```
(sv) [user@Ceagle-Plus-VM: sv-scripts]$ make tool-hello
PYTHONPATH=. benchexec -T 900 -t tool-hello --container sv-20170122-tool.xml
2017-01-23 21:58:04,661 - WARNING - Ignoring specified resource requirements in local-execution mode,
 only resource limits are used.

executing run set 'sv-comp17.tool-hello'     (1 file)
21:58:04   inv_square_int_true-unreach-call.c     true                            0.17    0.45

Statistics:                 1 Files
  correct:                  1
    correct true:           1
    correct false:          0
  incorrect:                0
    incorrect true:         0
    incorrect false:        0
  unknown:                  0
  Score:                    2 (max: 2)

In order to get HTML and CSV tables, run
table-generator 'results/sv-20170122-tool.2017-01-23_2158.results.sv-comp17.tool-hello.xml.bz2'
(sv) [user@Ceagle-Plus-VM: sv-scripts]$ _
```

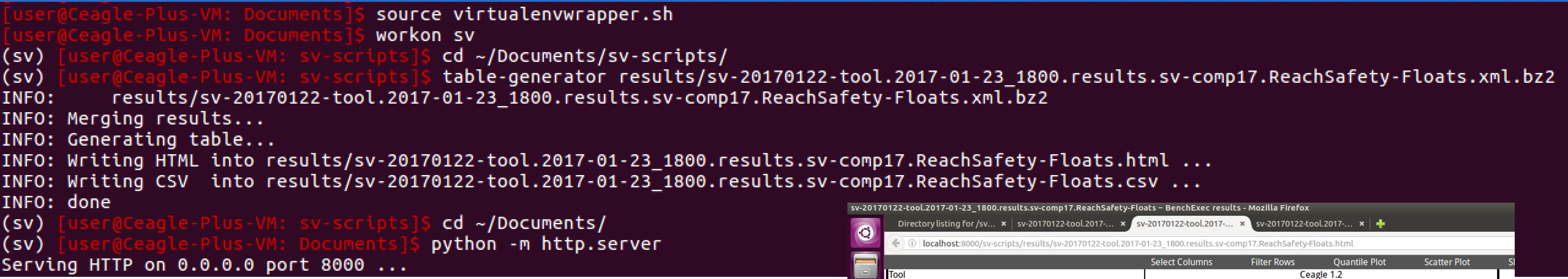
3.1 View benchmark results in a web browser

1. Open a Terminal app in Ceage-Plus-VM desktop GUI
2. Just type commands below **as-is** in the terminal:

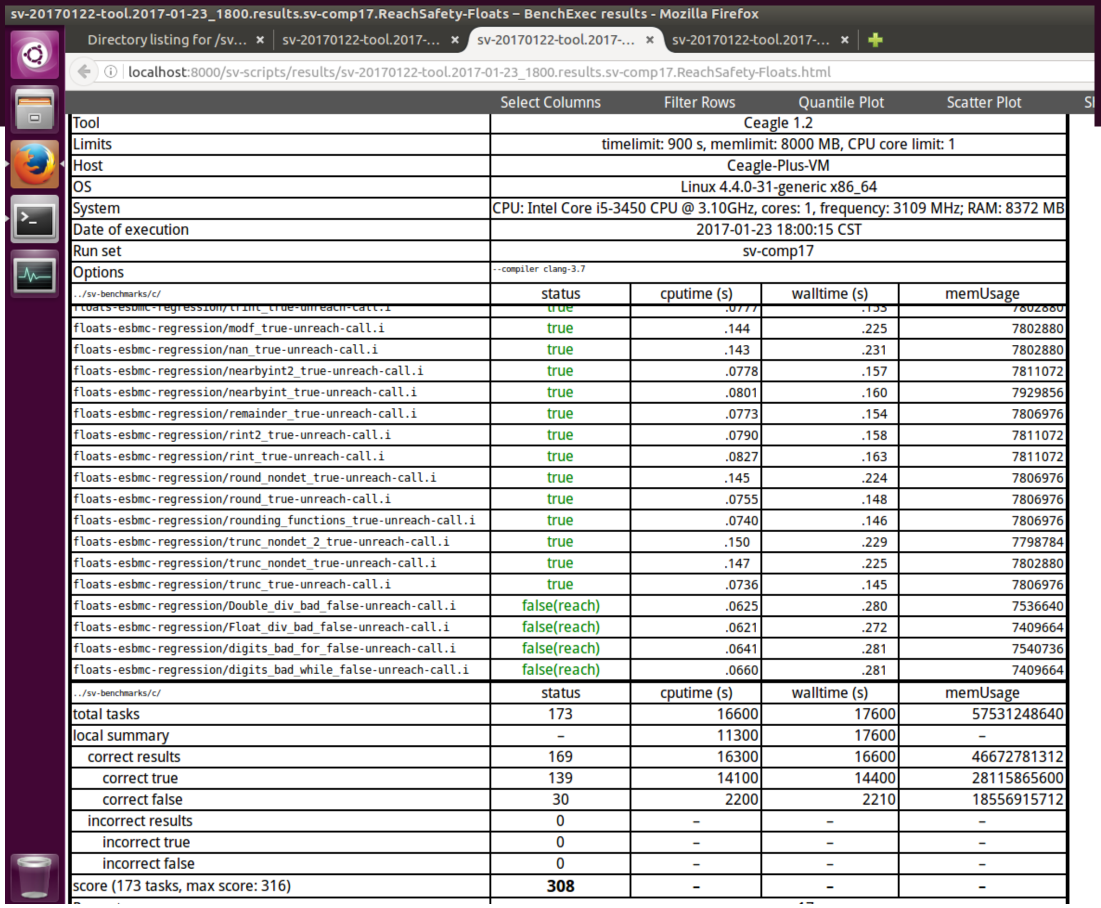
```
$ source virtualenvwrapper.sh
$ workon sv
(sv) $ cd ~/Documents/sv-scripts/
(sv) $ table-generator results/sv-20170122-tool.2017-01-23_1800.results.sv-
comp17.ReachSafety-Floats.xml.bz2
(sv) $ cd ~/Documents/
(sv) $ python -m http.server
```

3. Open a web browser and type the URL below:
   - http://localhost:8000/sv-scripts/results/
4. Select:
   `sv-20170122-tool.2017-01-23_1800.results.sv-comp17.ReachSafety-Floats.html`
5. The html can be used to view source code, log, memory, time, and so on.

3.2 Screen shot of 3.1 results



```
[user@Ceagle-Plus-VM: Documents]$ source virtualenvwrapper.sh
[user@Ceagle-Plus-VM: Documents]$ workon sv
(sv) [user@Ceagle-Plus-VM: sv-scripts]$ cd ~/Documents/sv-scripts/
(sv) [user@Ceagle-Plus-VM: sv-scripts]$ table-generator results/sv-20170122-tool.2017-01-23_1800.results.sv-comp17.ReachSafety-Floats.xml.bz2
INFO:     results/sv-20170122-tool.2017-01-23_1800.results.sv-comp17.ReachSafety-Floats.xml.bz2
INFO: Merging results...
INFO: Generating table...
INFO: Writing HTML into results/sv-20170122-tool.2017-01-23_1800.results.sv-comp17.ReachSafety-Floats.html ...
INFO: Writing CSV  into results/sv-20170122-tool.2017-01-23_1800.results.sv-comp17.ReachSafety-Floats.csv ...
INFO: done
(sv) [user@Ceagle-Plus-VM: sv-scripts]$ cd ~/Documents/
(sv) [user@Ceagle-Plus-VM: Documents]$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 ...
```



sv-20170122-tool.2017-01-23_1800.results.sv-comp17.ReachSafety-Floats – BenchExec results - Mozilla Firefox

localhost:8000/sv-scripts/results/sv-20170122-tool.2017-01-23_1800.results.sv-comp17.ReachSafety-Floats.html

Tool	Ceagle 1.2			
Limits	timelimit: 900 s, memlimit: 8000 MB, CPU core limit: 1			
Host	Ceagle-Plus-VM			
OS	Linux 4.4.0-31-generic x86_64			
System	CPU: Intel Core i5-3450 CPU @ 3.10GHz, cores: 1, frequency: 3109 MHz; RAM: 8372 MB			
Date of execution	2017-01-23 18:00:15 CST			
Run set	sv-comp17			
Options	--compiler clang-3.7			
../sv-benchmarks/c/	status	cputime (s)	walltime (s)	memUsage
floats-esbmc-regression/modf_true-unreach-call.i	true	.144	.225	7802880
floats-esbmc-regression/nan_true-unreach-call.i	true	.143	.231	7802880
floats-esbmc-regression/nearbyint2_true-unreach-call.i	true	.0778	.157	7811072
floats-esbmc-regression/nearbyint_true-unreach-call.i	true	.0801	.160	7929856
floats-esbmc-regression/remainder_true-unreach-call.i	true	.0773	.154	7806976
floats-esbmc-regression/rint2_true-unreach-call.i	true	.0790	.158	7811072
floats-esbmc-regression/rint_true-unreach-call.i	true	.0827	.163	7811072
floats-esbmc-regression/round_nondet_true-unreach-call.i	true	.145	.224	7806976
floats-esbmc-regression/round_true-unreach-call.i	true	.0755	.148	7806976
floats-esbmc-regression/rounding_functions_true-unreach-call.i	true	.0740	.146	7806976
floats-esbmc-regression/trunc_nondet_2_true-unreach-call.i	true	.150	.229	7798784
floats-esbmc-regression/trunc_nondet_true-unreach-call.i	true	.147	.225	7802880
floats-esbmc-regression/trunc_true-unreach-call.i	true	.0736	.145	7806976
floats-esbmc-regression/Double_div_bad_false-unreach-call.i	false(reach)	.0625	.280	7536640
floats-esbmc-regression/Float_div_bad_false-unreach-call.i	false(reach)	.0621	.272	7409664
floats-esbmc-regression/digits_bad_for_false-unreach-call.i	false(reach)	.0641	.281	7540736
floats-esbmc-regression/digits_bad_while_false-unreach-call.i	false(reach)	.0660	.281	7409664
../sv-benchmarks/c/	status	cputime (s)	walltime (s)	memUsage
total tasks	173	16600	17600	57531248640
local summary	–	11300	17600	–
correct results	169	16300	16600	46672781312
correct true	139	14100	14400	28115865600
correct false	30	2200	2210	18556915712
incorrect results	0	–	–	–
incorrect true	0	–	–	–
incorrect false	0	–	–	–
score (173 tasks, max score: 316)	**308**	-	-	-

3.3 benchexec benchmarking in detail

1. All files needed for Ceagle+ benchmarking are stored at:
   - ~/Documents/sv-scripts/
     - Makefile: make scripts for performing preset benchmarks
       - preset benchmarks include: tool-hello, tool-experiments, tool-examples
     - sv-20170122-tool.xml: main configuration file as an input to benchexec
       - specifies verification files
         - in tags: <includesfile> ... </includesfile>
       - specifies verification property
         - in tags: <propertyfile> ... </propertyfile>
     - results/: a folder used by benchexec to generate benchmarking results
       - please use commands mentions in 3.1 & 3.2 to view results in a more structured view
     - fix.sh: a bash script that enables benchexec to measure CPU time and memory usage
       - this script should be executed in every new terminal before benchmarking
2. Benchmarks are stored at:
   - ~/Documents/sv-benchmarks/
   - ~/Documents/sv-examples/
3. You can customize files in sv-scripts/sv-benchmarks/sv-examples to perform benchmarks on your own.

3.4 About fix.sh

- Content of fix.sh:
  - sudo chmod o+wt '/sys/fs/cgroup/cpu,cpuacct/'
  - sudo chmod o+wt '/sys/fs/cgroup/freezer/'
  - sudo chmod o+wt '/sys/fs/cgroup/memory/'
  - sudo chmod o+wt '/sys/fs/cgroup/cpuset/'
  - sudo swapoff -a
- Why it's important?
  - doing so is required by benchexec to measure CPU time and memory usage
  - this script should be executed in every new terminal before benchmarking
- Help about benchexec:
  - benchexec is used to measure accurate time & memory usage
  - Website: https://github.com/sosy-lab/benchexec
  - Manual: https://github.com/sosy-lab/benchexec/blob/master/doc/INDEX.md

4. Reproduce the experiments of Ceagle+

1. Open a Terminal app in Ceagle-Plus-VM desktop GUI
2. Just type commands below **as-is** in the terminal:

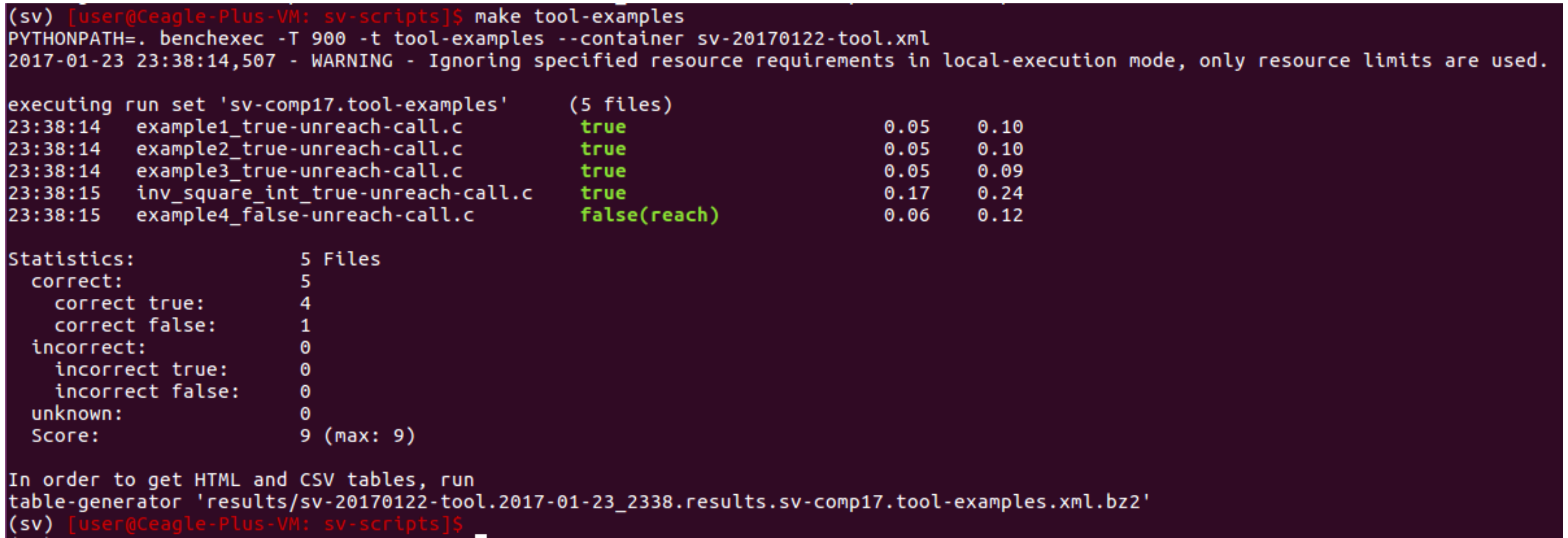
```
$ cd ~/Documents/sv-scripts/
$ source virtualenvwrapper.sh
$ workon sv
(sv) $ ./fix.sh
(sv) $ export PATH=$PATH:~/Downloads/ceagle-plus/
(sv) $ make tool-experiments
```

3. Caution:
   - The overall experiments procedure may take hours to complete.
   - Due to hardware limitations of VM build, experiment results may be different to the ones in paper.

5. Try different tasks on your own

1. Open a Terminal app in Ceagle-Plus-VM desktop GUI
2. Just type commands below **as-is** in the terminal:

```
$ cd ~/Documents/sv-scripts/
$ source virtualenvwrapper.sh
$ workon sv
(sv) $ ./fix.sh
(sv) $ export PATH=$PATH:~/Downloads/ceagle-plus/
(sv) $ make tool-examples
```



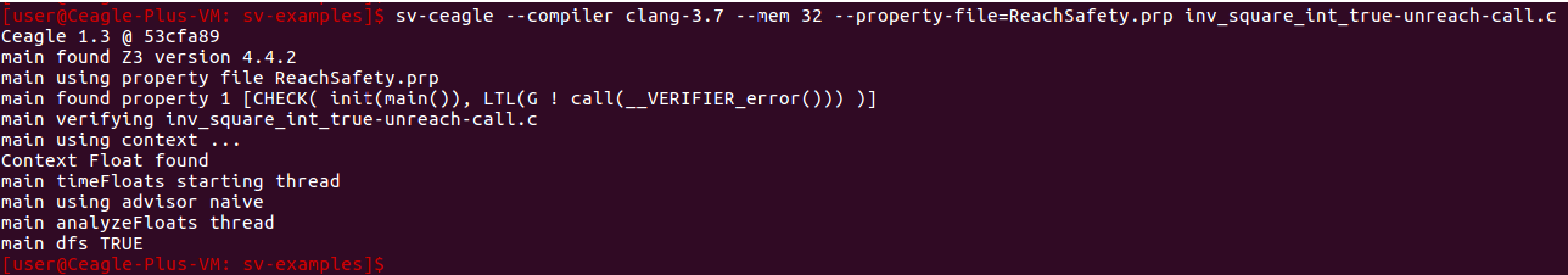
```
(sv) [user@Ceagle-Plus-VM: sv-scripts]$ make tool-examples
PYTHONPATH=. benchexec -T 900 -t tool-examples --container sv-20170122-tool.xml
2017-01-23 23:38:14,507 - WARNING - Ignoring specified resource requirements in local-execution mode, only resource limits are used.

executing run set 'sv-comp17.tool-examples'     (5 files)
23:38:14   example1_true-unreach-call.c          true                 0.05    0.10
23:38:14   example2_true-unreach-call.c          true                 0.05    0.10
23:38:14   example3_true-unreach-call.c          true                 0.05    0.09
23:38:15   inv_square_int_true-unreach-call.c    true                 0.17    0.24
23:38:15   example4_false-unreach-call.c         false(reach)         0.06    0.12

Statistics:             5 Files
  correct:              5
    correct true:       4
    correct false:      1
  incorrect:            0
    incorrect true:     0
    incorrect false:    0
  unknown:              0
  Score:                9 (max: 9)

In order to get HTML and CSV tables, run
table-generator 'results/sv-20170122-tool.2017-01-23_2338.results.sv-comp17.tool-examples.xml.bz2'
(sv) [user@Ceagle-Plus-VM: sv-scripts]$
```

6. Run Ceagle+ by hand

1. Open a Terminal app in Ceagle-Plus-VM desktop GUI
2. Try a single file verification by typing in the terminal:

```
$ export PATH=$PATH:~/Downloads/ceagle-plus/
$ cd ~/Documents/sv-examples/
$ sv-ceagle --compiler clang-3.7 --mem 32 --property-file=ReachSafety.prp
inv_square_int_true-unreach-call.c
```

3. More benchmark files can be found at:
   - ~/Downloads/sv-examples
   - ~/Downloads/sv-benchmarks
4. Counter-example file can be found at witness.graphml



```
[user@Ceagle-Plus-VM: sv-examples]$ sv-ceagle --compiler clang-3.7 --mem 32 --property-file=ReachSafety.prp inv_square_int_true-unreach-call.c
Ceagle 1.3 @ 53cfa89
main found Z3 version 4.4.2
main using property file ReachSafety.prp
main found property 1 [CHECK( init(main()), LTL(G ! call(__VERIFIER_error())) )]
main verifying inv_square_int_true-unreach-call.c
main using context ...
Context Float found
main timeFloats starting thread
main using advisor naive
main analyzeFloats thread
main dfs TRUE
[user@Ceagle-Plus-VM: sv-examples]$
```

7. Craft a benchmark of your own

1. A verification benchmark is a little different to normal source file
   - a property should be inserted to source file to verify it
   - an assumption is optional, to specify a program entry state
   - the file name should be modified to indicate the expected result
     - benchexec uses filenames to determine whether the verifier reported results correctly
2. Insert assumption (optional)
   - use __VERIFIER_assume(bool exp) to insert assumptions
3. Insert property
   - use __VERIFIER_assert(bool exp) & __VERIFIER_error(bool exp) to insert properties
4. Change file name, say the original file name is hello.c
   - if the inserted property is expected to be TRUE, change to hello_true-unreach-call.c
   - if the inserted property is expected to be FALSE, change to hello_false-unreach-call.c
   - the example file hello_false-unreach-call.c is shown right, refer to Sec. 6 to verify it.

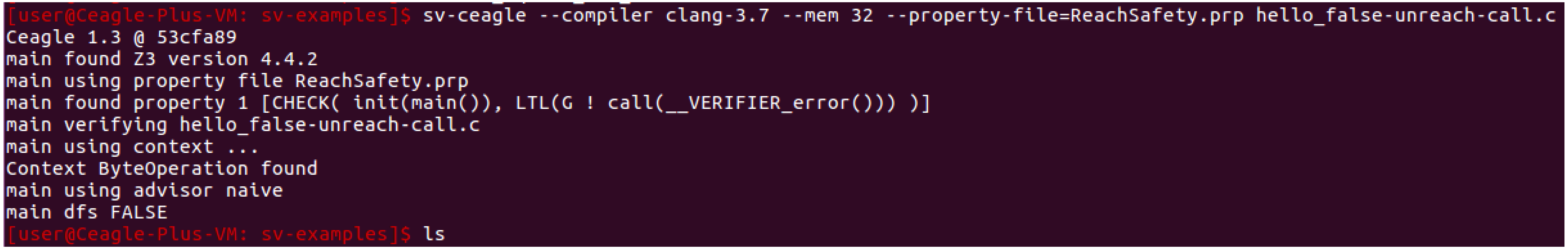
7.1 Verify hello_false-unreach-call.c

```c
void __VERIFIER_assert(int cond) {
  if (!(cond)) {
ERROR: __VERIFIER_error();
  }
  return;
}

int main() {
  int a;
  __VERIFIER_assume(a == 0);
  __VERIFIER_assert(a != 0);
  return 0;
}
```

1. Open a Terminal app in Ceagle-Plus-VM desktop GUI
2. Try a single file verification by typing in the terminal:

```
$ export PATH=$PATH:~/Downloads/ceagle-plus/
$ cd ~/Documents/sv-examples/
$ sv-ceagle --compiler clang-3.7 --mem 32 --property-file=ReachSafety.prp
hello_false-unreach-call.c
```

3. More benchmark files can be found at:
   - ~/Downloads/sv-examples
   - ~/Downloads/sv-benchmarks
4. Counter-example file can be found at witness.graphml



```
[user@Ceagle-Plus-VM: sv-examples]$ sv-ceagle --compiler clang-3.7 --mem 32 --property-file=ReachSafety.prp hello_false-unreach-call.c
Ceagle 1.3 @ 53cfa89
main found Z3 version 4.4.2
main using property file ReachSafety.prp
main found property 1 [CHECK( init(main()), LTL(G ! call(__VERIFIER_error())) )]
main verifying hello_false-unreach-call.c
main using context ...
Context ByteOperation found
main using advisor naive
main dfs FALSE
[user@Ceagle-Plus-VM: sv-examples]$ ls
```

The end