demo

July 27, 2020

1 Demo for functional usage of CFG-explorer

Now, `cfg-explorer` can not only be used as a command line tool. We can also call it within a Python program.

1.1 Download Spec CPU Benchmark 2006

Save the suite outside our current repository:

```
$ cd ..
$ git clone https://github.com/Multi2Sim/m2s-bench-spec2006
```

Every `.i386` file is a binary file for testing.

1.2 Import Libraries

First, to import `cfg_explore` in this subdirectory, you should include your `cfg-explorer` path into your `PATH` environment variable. You can do this by (suppose the whole `cfg-explorer` directory is located in `$HOME/cfg-explorer`):

```
$ export PATH=$HOME/cfg-explorer:$PATH
```

Or import it in this notebook by such an approach for that the target folder is actually the parent folder of this file:

[1]:
```python
import os
import sys
from pathlib import Path

sys.path.insert(0,str(Path().resolve().parent))
```

1.3 Usages of `cfg_explore` Function

[2]:
```python
from cfgexplorer import cfg_explore
```

1.3.1 Lanuch an interactive web app

Now, call `cfg_explore` with the only argument `binary`, which is the path of the bianry file we prepare to analysis. After running, it will host a website on http://127.0.0.1:5050/ to show the *control flow graph* of the file. You can specify the port by `port` parameter.

```
[3]: cfg_explore(binary='../../m2s-bench-spec2006/999.specrand/specrand_base.i386')
```

```
WARNING | 2020-07-27 00:05:27,559 | angr.analyses.cfg.cfg_fast |
"collect_data_references" is deprecated and will be removed soon. Please

use "data_references" instead
100% |####################################| Elapsed Time: 0:00:00 Time:  0:00:00

 * Serving Flask app "cfgexplorer.explorer" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production

deployment.
   Use a production WSGI server instead.
 * Debug mode: on

INFO    | 2020-07-27 00:05:27,834 | werkzeug |  * Running on

http://127.0.0.1:5050/ (Press CTRL+C to quit)
INFO    | 2020-07-27 00:06:00,927 | werkzeug | 127.0.0.1 - -

[27/Jul/2020 00:06:00] "GET / HTTP/1.1" 200 -
INFO    | 2020-07-27 00:06:01,059 | werkzeug | 127.0.0.1 - -

[27/Jul/2020 00:06:01] "GET /js/svg-pan-zoom.js HTTP/1.1" 200 -
INFO    | 2020-07-27 00:06:03,382 | werkzeug | 127.0.0.1 - -

[27/Jul/2020 00:06:03] "GET /api/cfg/0x80483a0 HTTP/1.1" 200 -
INFO    | 2020-07-27 00:06:03,554 | werkzeug | 127.0.0.1 - -

[27/Jul/2020 00:06:03] "GET /favicon.ico HTTP/1.1" 404 -
```

Whenever you want to shut down the app, just interrupt the function. For example, in this notebook, click on interrupt the kernelbuttom on the toolbar.

1.3.2 Export raw .dot files

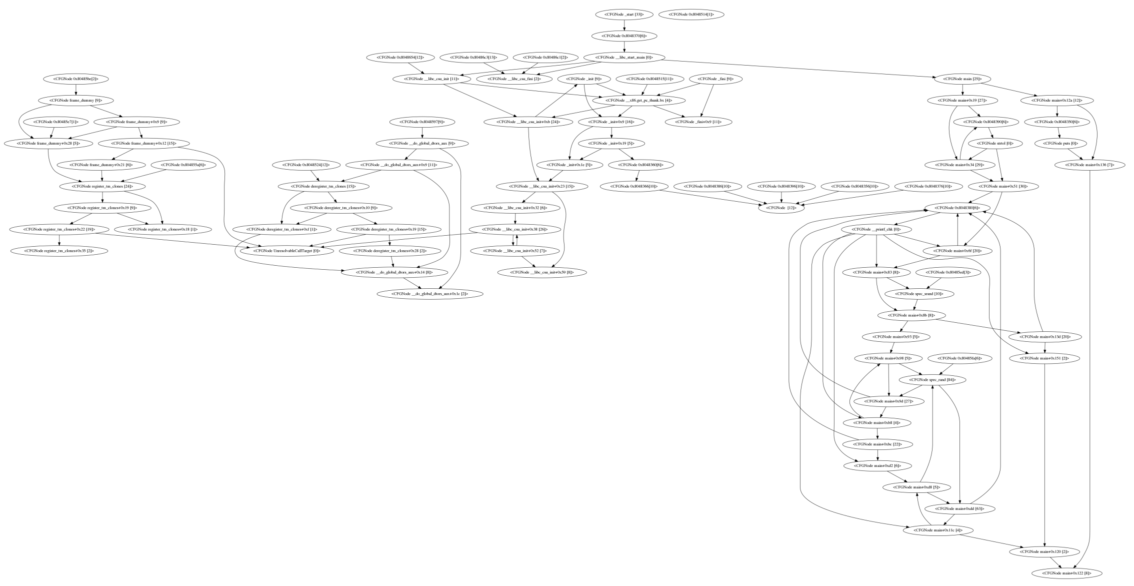
```
[5]: cfg_explore(binary='../../m2s-bench-spec2006/999.specrand/specrand_base.
     ↪i386',output='test.dot')
```

```
WARNING | 2020-07-27 00:08:49,522 | angr.analyses.cfg.cfg_fast |
"collect_data_references" is deprecated and will be removed soon. Please

use "data_references" instead
100% |####################################| Elapsed Time: 0:00:00 Time:  0:00:00
```

.dot file can be converted to image format, for example, if you have installed graphviz in your machine, try this command:

[6]:
```
!dot test.dot -Tpng -o test.png
```



1.3.3 Export .svg files

You can also specify the output argument with .svg suffix, and you will get the same graph as what you see in the web app without output Section 1.3.1

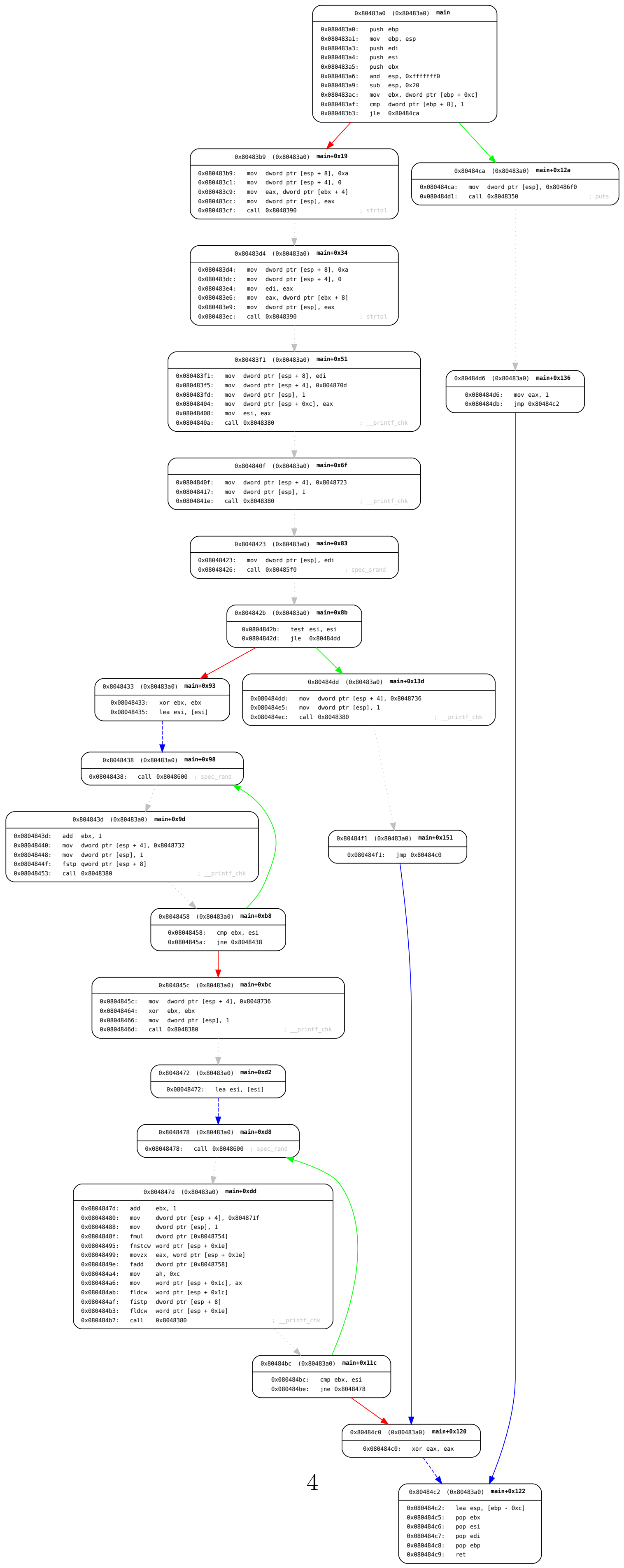
[7]:
```
cfg_explore(binary='../../m2s-bench-spec2006/999.specrand/specrand_base.
↪i386',output='./test.svg')
```

```
WARNING | 2020-07-27 00:09:00,370 | angr.analyses.cfg.cfg_fast |
"collect_data_references" is deprecated and will be removed soon. Please

use "data_references" instead
100% |######################################| Elapsed Time: 0:00:00 Time:  0:00:00
```

.svg files can be opened by web-browser directly, it can also be displayed in this notebook:

[8]:
```
from IPython.core.display import SVG
display(SVG('test.svg'))
```



```
0x80483a0  (0x80483a0)  main
0x080483a0:   push ebp
0x080483a1:   mov  ebp, esp
0x080483a3:   push edi
0x080483a4:   push esi
0x080483a5:   push ebx
0x080483a6:   and  esp, 0xfffffff0
0x080483a9:   sub  esp, 0x20
0x080483ac:   mov  ebx, dword ptr [ebp + 0xc]
0x080483af:   cmp  dword ptr [ebp + 8], 1
0x080483b3:   jle  0x80484ca
```

```
0x80483b9  (0x80483a0)  main+0x19
0x080483b9:   mov  dword ptr [esp + 8], 0xa
0x080483c1:   mov  dword ptr [esp + 4], 0
0x080483c9:   mov  eax, dword ptr [ebx + 4]
0x080483cc:   mov  dword ptr [esp], eax
0x080483cf:   call 0x8048390                ; strtol
```

```
0x80484ca  (0x80483a0)  main+0x12a
0x080484ca:   mov  dword ptr [esp], 0x80486f0
0x080484d1:   call 0x8048350                        ; puts
```

```
0x80483d4  (0x80483a0)  main+0x34
0x080483d4:   mov  dword ptr [esp + 8], 0xa
0x080483dc:   mov  dword ptr [esp + 4], 0
0x080483e4:   mov  edi, eax
0x080483e6:   mov  eax, dword ptr [ebx + 8]
0x080483e9:   mov  dword ptr [esp], eax
0x080483ec:   call 0x8048390                ; strtol
```

```
0x80483f1  (0x80483a0)  main+0x51
0x080483f1:   mov  dword ptr [esp + 8], edi
0x080483f5:   mov  dword ptr [esp + 4], 0x804870d
0x080483fd:   mov  dword ptr [esp], 1
0x08048404:   mov  dword ptr [esp + 0xc], eax
0x08048408:   mov  esi, eax
0x0804840a:   call 0x8048380                        ; __printf_chk
```

```
0x80484d6  (0x80483a0)  main+0x136
0x080484d6:   mov eax, 1
0x080484db:   jmp 0x80484c2
```

```
0x804840f  (0x80483a0)  main+0x6f
0x0804840f:   mov  dword ptr [esp + 4], 0x8048723
0x08048417:   mov  dword ptr [esp], 1
0x0804841e:   call 0x8048380                        ; __printf_chk
```

```
0x8048423  (0x80483a0)  main+0x83
0x08048423:   mov  dword ptr [esp], edi
0x08048426:   call 0x80485f0            ; spec_srand
```

```
0x804842b  (0x80483a0)  main+0x8b
0x0804842b:   test esi, esi
0x0804842d:   jle  0x80484dd
```

```
0x8048433  (0x80483a0)  main+0x93
0x08048433:   xor ebx, ebx
0x08048435:   lea esi, [esi]
```

```
0x80484dd  (0x80483a0)  main+0x13d
0x080484dd:   mov  dword ptr [esp + 4], 0x8048736
0x080484e5:   mov  dword ptr [esp], 1
0x080484ec:   call 0x8048380                        ; __printf_chk
```

```
0x8048438  (0x80483a0)  main+0x98
0x08048438:   call 0x8048600  ; spec_rand
```

```
0x804843d  (0x80483a0)  main+0x9d
0x0804843d:   add  ebx, 1
0x08048440:   mov  dword ptr [esp + 4], 0x8048732
0x08048448:   mov  dword ptr [esp], 1
0x0804844f:   fstp qword ptr [esp + 8]
0x08048453:   call 0x8048380                        ; __printf_chk
```

```
0x80484f1  (0x80483a0)  main+0x151
0x080484f1:   jmp 0x80484c0
```

```
0x8048458  (0x80483a0)  main+0xb8
0x08048458:   cmp ebx, esi
0x0804845a:   jne 0x8048438
```

```
0x804845c  (0x80483a0)  main+0xbc
0x0804845c:   mov  dword ptr [esp + 4], 0x8048736
0x08048464:   xor  ebx, ebx
0x08048466:   mov  dword ptr [esp], 1
0x0804846d:   call 0x8048380                        ; __printf_chk
```

```
0x8048472  (0x80483a0)  main+0xd2
0x08048472:   lea esi, [esi]
```

```
0x8048478  (0x80483a0)  main+0xd8
0x08048478:   call 0x8048600  ; spec_rand
```

```
0x804847d  (0x80483a0)  main+0xdd
0x0804847d:   add    ebx, 1
0x08048480:   mov    dword ptr [esp + 4], 0x804871f
0x08048488:   mov    dword ptr [esp], 1
0x0804848f:   fmul   dword ptr [0x8048754]
0x08048495:   fnstcw word ptr [esp + 0x1e]
0x08048499:   movzx  eax, word ptr [esp + 0x1e]
0x0804849e:   fadd   dword ptr [0x8048758]
0x080484a4:   mov    ah, 0xc
0x080484a6:   mov    word ptr [esp + 0x1c], ax
0x080484ab:   fldcw  word ptr [esp + 0x1c]
0x080484af:   fistp  dword ptr [esp + 8]
0x080484b3:   fldcw  word ptr [esp + 0x1e]
0x080484b7:   call   0x8048380                      ; __printf_chk
```

```
0x80484bc  (0x80483a0)  main+0x11c
0x080484bc:   cmp ebx, esi
0x080484be:   jne 0x8048478
```

```
0x80484c0  (0x80483a0)  main+0x120
0x080484c0:   xor eax, eax
```

```
0x80484c2  (0x80483a0)  main+0x122
0x080484c2:   lea esp, [ebp - 0xc]
0x080484c5:   pop ebx
0x080484c6:   pop esi
0x080484c7:   pop edi
0x080484c8:   pop ebp
0x080484c9:   ret
```

There are also many online tools available that convert `.svg` to other format files. Besides, if you have installed `inkscape`, you can use:

```
[9]: !inkscape test.svg --export-area-drawing --without-gui --export-pdf=test.pdf
```

```
Failed to get connection
** (inkscape:18020): CRITICAL **: 00:09:21.504:
dbus_g_proxy_new_for_name: assertion 'connection != NULL' failed

** (inkscape:18020): CRITICAL **: 00:09:21.504:
dbus_g_proxy_call: assertion 'DBUS_IS_G_PROXY (proxy)' failed

** (inkscape:18020): CRITICAL **: 00:09:21.504:
dbus_g_connection_register_g_object: assertion 'connection != NULL' failed
```

And now, you can open `test.pdf` directly to view the *control flow graph*. It is what TEX exactly do when asked to insert a `.svg` image into an article by `\includegraphics{}`. It is to say that, if `inkscape`and TEX installed properly, this notebook can be converted to a pretty pdf by `nbconvert`, which is built-in Jupyter notebook server.

1.3.4 Traversal a large folder to generate all CFGs

We still use `m2s-bench-spec2006` as an example.

Assume that we need to analyze all binary files in this folder. Wrapping `cfg-explorer` as a function makes the task more flexible inside a Python script.

First, get all potential binary files for analysis:

```
[10]: from glob import glob
      progs = sorted(glob('../../m2s-bench-spec2006/*/*.i386'))
      progs
```

```
[10]: ['../../m2s-bench-spec2006/401.bzip2/bzip2_base.i386',
       '../../m2s-bench-spec2006/403.gcc/gcc_base.i386',
       '../../m2s-bench-spec2006/410.bwaves/bwaves_base.i386',
       '../../m2s-bench-spec2006/416.gamess/gamess_base.i386',
       '../../m2s-bench-spec2006/429.mcf/mcf_base.i386',
       '../../m2s-bench-spec2006/433.milc/milc_base.i386',
       '../../m2s-bench-spec2006/434.zeusmp/zeusmp_base.i386',
       '../../m2s-bench-spec2006/435.gromacs/gromacs_base.i386',
       '../../m2s-bench-spec2006/436.cactusADM/cactusADM_base.i386',
       '../../m2s-bench-spec2006/437.leslie3d/leslie3d_base.i386',
       '../../m2s-bench-spec2006/444.namd/namd_base.i386',
       '../../m2s-bench-spec2006/445.gobmk/gobmk_base.i386',
       '../../m2s-bench-spec2006/447.dealII/dealII_base.i386',
       '../../m2s-bench-spec2006/450.soplex/soplex_base.i386',
       '../../m2s-bench-spec2006/453.povray/povray_base.i386',
       '../../m2s-bench-spec2006/454.calculix/calculix_base.i386',
       '../../m2s-bench-spec2006/456.hmmer/hmmer_base.i386',
       '../../m2s-bench-spec2006/458.sjeng/sjeng_base.i386',
       '../../m2s-bench-spec2006/459.GemsFDTD/GemsFDTD_base.i386',
       '../../m2s-bench-spec2006/462.libquantum/libquantum_base.i386',
       '../../m2s-bench-spec2006/464.h264ref/h264ref_base.i386',
       '../../m2s-bench-spec2006/465.tonto/tonto_base.i386',
       '../../m2s-bench-spec2006/470.lbm/lbm_base.i386',
       '../../m2s-bench-spec2006/471.omnetpp/omnetpp_base.i386',
       '../../m2s-bench-spec2006/473.astar/astar_base.i386',
       '../../m2s-bench-spec2006/481.wrf/wrf_base.i386',
       '../../m2s-bench-spec2006/482.sphinx3/sphinx3_base.i386',
       '../../m2s-bench-spec2006/483.xalancbmk/xalancbmk_base.i386',
       '../../m2s-bench-spec2006/998.specrand/specrand_base.i386',
       '../../m2s-bench-spec2006/999.specrand/specrand_base.i386']
```

[11]:
```
# create a directory to store the outputs
out_dir = './output'
if not os.path.exists(out_dir):
    os.mkdir(out_dir)
```

Then, we can simply call `cfg_explore` function inside loops. Keep it alone, we just need to wait for generating all '.svg' files.

Note: it might take a extremely long time. Be patient.

[12]:
```
for p in progs:
    name = p.split('/')[3]
    print('start analysis of:',name)
    output_file = os.path.join(out_dir, name + '.svg')
    if not os.path.exists(output_file):
        cfg_explore(binary=p,output=output_file)
```

```
start analysis of: 401.bzip2
start analysis of: 403.gcc
start analysis of: 410.bwaves
start analysis of: 416.gamess
start analysis of: 429.mcf
start analysis of: 433.milc
start analysis of: 434.zeusmp
start analysis of: 435.gromacs
start analysis of: 436.cactusADM
start analysis of: 437.leslie3d
start analysis of: 444.namd
start analysis of: 445.gobmk
start analysis of: 447.dealII
start analysis of: 450.soplex
start analysis of: 453.povray
start analysis of: 454.calculix
start analysis of: 456.hmmer
start analysis of: 458.sjeng
start analysis of: 459.GemsFDTD
start analysis of: 462.libquantum
start analysis of: 464.h264ref
start analysis of: 465.tonto
start analysis of: 470.lbm
start analysis of: 471.omnetpp
start analysis of: 473.astar
start analysis of: 481.wrf
start analysis of: 482.sphinx3
start analysis of: 483.xalancbmk
start analysis of: 998.specrand
start analysis of: 999.specrand
```

Now, you can view all outputs in `out_dir`.