

Linux Verification Project

The objectives of this part of the project are:

- Explore the complexities that can arise while verifying correct pairing of spin lock with spin unlock.
 - Evaluate the quality of the given evidence
1. Explore the *Inter-Procedural Complexity* to find:
 - a. Spin lock instances for which MPG is bigger than one.
 - b. Spin lock instances with small, medium, and large values for MPG
 - c. Spin Lock instances that you can verify manually using the given evidence
 - d. Spin Lock instances that you cannot verify manually using the given evidence

Section 2: Report on 10 interesting instances you have found. Write a short description of each instance. Comment on the quality of the evidence given for each instance. The instances should present a good variety of interesting complexities.

2. Explore the *Intra-Procedural Complexity* to find:
 - a. Spin lock instances for which MPG = 1
 - b. Spin lock instances with interesting variations based on CFG(n), CFG(e), EFG(n), and EFG(e).
 - c. Spin lock instances requiring path feasibility check.

Section 3: Report on 10 interesting instances you have found. Write a short description of each instance. Comment on the quality of the evidence given for each instance. The instances should present a good variety of interesting complexities.

Section 1: Write an introduction to highlight your major findings.

Section 4: Write a conclusion to tell what other interesting things you would do if you were to follow up on your exploration.

Resources to help with the exploration:

- A website of different instances, with associated graphs for MPG, EFG, and CFG
- A spreadsheet to summarize the information on the website (<http://kcs1.ece.iastate.edu/linux-results/>).
- You should get the relevant Linux source code.

You can browse the Linux source code using the *source link*: <http://lxr.free-electrons.com/source/>

You can use this website with the given spreadsheet as follows.

Suppose you want to view the source for `goku_irq` instance.

1. The spreadsheet gives the key - [8029+22+/linux-3.19-rc1/drivers/usb/gadget/udc/goku_udc.c]
2. Remove the id (8029+22) and copy the path after linux-3.19-rc1. In this case, you will copy `drivers/usb/gadget/udc/goku_udc.c`
3. Concatenate with the source link. In this case, you get http://lxr.free-electrons.com/source/drivers/usb/gadget/udc/goku_udc.c
4. If you don't specify version you will be directed to the latest kernel version. So add `?v=3.19` at the end. For kernel version 3.19, the url would be:
http://lxr.free-electrons.com/source/drivers/usb/gadget/udc/goku_udc.c?v=3.19

The website and the spreadsheet contain evidence for verifying the correct pairing. Comment on the quality of the evidence with respect to following questions:

- A. Is MPG the *necessary* set of functions, i.e., are there LOCK instances where the MPG includes functions that are not necessary for completing the pairing analysis?
- B. Is MPG the *sufficient* set of functions, i.e., are there LOCK instances where the MPG misses some functions that are necessary for completing the pairing analysis?

- C. Is EFG easier compared to CFG to understand the pairing?
- D. Is it easy to derive EFG from CFG by hand?
- E. Is EFG *necessary*, i.e., are there cases where some nodes or edges in the EFG not necessary for the pairing analysis?
- F. Is EFG *sufficient*, i.e., are there cases EFG misses some nodes or edges from the CFG that are necessary for completing the pairing analysis?