Coccinelle: A program matching and transformation tool

Himangi Saraogi, Linux kernel intern,
FOSS Outreach Program for Women Round 8,
Mentor: Julia Lawall

Linux.conf.au

Literally

A Coccinelle (ladybug) is a bug that eats smaller bugs.

My work with Coccinelle!

Develop/harden coccinelle semantic patches to integrate into the kernel.

- Identify bugs that are prevalent across the kernel. (coccinellery)
- Send patches solving the bug to discuss whether it is an issue of concern.
- Develop coccinelle scripts to fix those bugs.
- Analyze results of the scripts.
- Send patches for the scripts to be accepted into the kernel.

Why do we need Coccinelle?

- Bugs are unfortunate but everywhere.
- Systems code is often huge and rapidly evolving.
- Systems code is often in C.
- Linux is a highly critical software with a huge codebase.
- There are various developers with different levels of experience contributing to the kernel.

Common programming problems

- Programmers don’t really understand how C works.
  - !e1 & e2 does a bit-and with 0 or 1.
- A simpler API function exists, but not everyone uses it.
  - Mixing different functions for the same purpose is confusing.
- A function may fail, but the call site doesn’t check for that.
  - A rare error case will cause an unexpected crash
- Etc.

Need for pervasive code changes

Example: Bad bit-and

```
if (!dma_cntrl & DMA_START_BIT) {
    BCMLOG(BCMLOG_DBG, "Already Stopped\n");
    return BC_STS_SUCCESS;
}
```

From drivers/staging/crystalhd/crystalhd hw.c

Example: Inconsistent API usage

drivers/mtd/nand/r852.c:

```
if (!bounce) {
  dev->phys_dma_addr =
    pci_map_single(dev->pci_dev, (void *)buf, R852_DMA_LEN,
      (do_read ? PCI_DMA_FROMDEVICE : PCI_DMA_TODEVICE));

  if (pci_dma_mapping_error(dev->pci_dev, dev->phys_dma_addr))
    bounce = 1;
}
```

drivers/mtd/nand/denali.c:

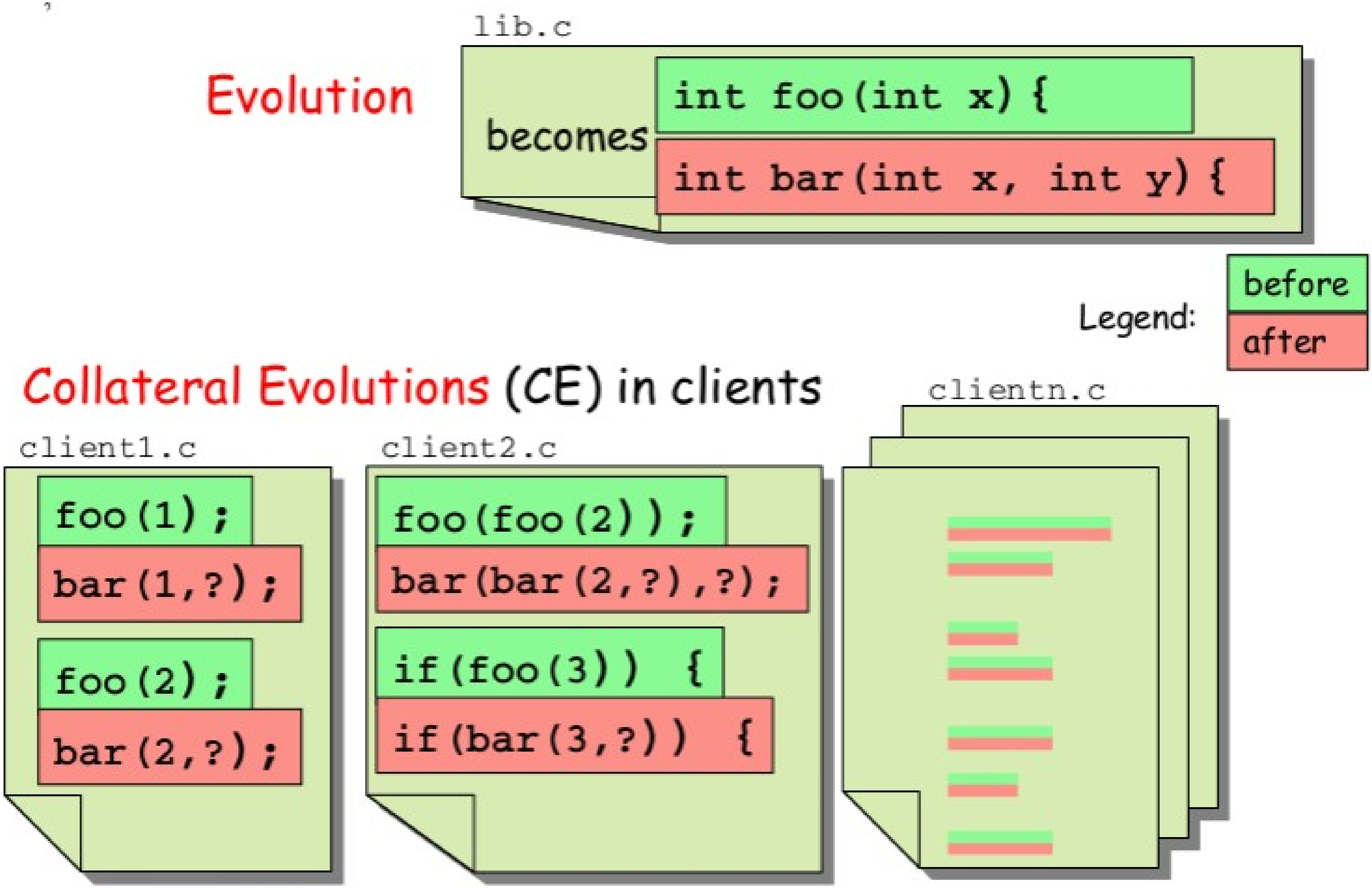
```
denali->buf.dma_buf =
   dma_map_single(&dev->dev, denali->buf.buf, DENALI_BUF_SIZE,
                  DMA_BIDIRECTIONAL);
if (dma_mapping_error(&dev->dev, denali->buf.dma_buf))  ...
pci_set_master(dev);
...
ret = pci_request_regions(dev, DENALI_NAND_NAME);
```

Example: Missing error check

```
alloc = kmalloc(sizeof *alloc, GFP_KERNEL);
INIT_LIST_HEAD(&intmem_allocations);
intmem_virtual = ioremap(MEM_INTMEM_START + RESERVED_SIZE,
                         MEM_INTMEM_SIZE - RESERVED_SIZE);
initiated = 1;
alloc->size = MEM_INTMEM_SIZE - RESERVED_SIZE;
```

From arch/cris/arch-v32/mm/intmem.c

Collateral Evolutions



Why is collateral evolution significant?

- The kernel has many libraries each with many clients.
  - Lots of driver support libraries: one per device type, one per bus (pci library, sound library, ...).
  - Lots of device specific code : Drivers make up more than 50% of Linux.
- Many **evolutions** and **collateral evolutions** occur.
- Examples of evolution :
  - Add argument, split data structure, getter and setter introduction, protocol change, change return type, add error checking, ...

Requirements for automation

- The ability to abstract over irrelevant information:
  - if (!dma_cntrl & DMA START BIT) { ... }: dma_cntrl is not important.
- The ability to match scattered code fragments:
  - kmalloc may be far from the first dereference.
- The ability to transform code fragments:
  - Replace pci map single by dma map single, or vice versa.

Our goals

- Bug finding and fixing
  - Automatically find code containing bugs or defects.
  - Automatically fix bugs or defects.
  - Provide a system that is accessible to software developers.
- Collateral evolutions
  - Search for patterns of interaction with the library
  - Systematically transform the interaction code

What Coccinelle can do?

- Static analysis to find patterns in C source code.
- Automatic transformation to fix bugs.
- Generate different information of bugs based on script mode.
  - Patch : apply transformations to files where the bug is detected.
  - Context : just marks out the changes that will be done, without actually making the changes.
  - Org : lists in TODO format with exact line number and column positions of the bugs.
  - Report : logs a custom message which has the line numbers and files with the warning or error.

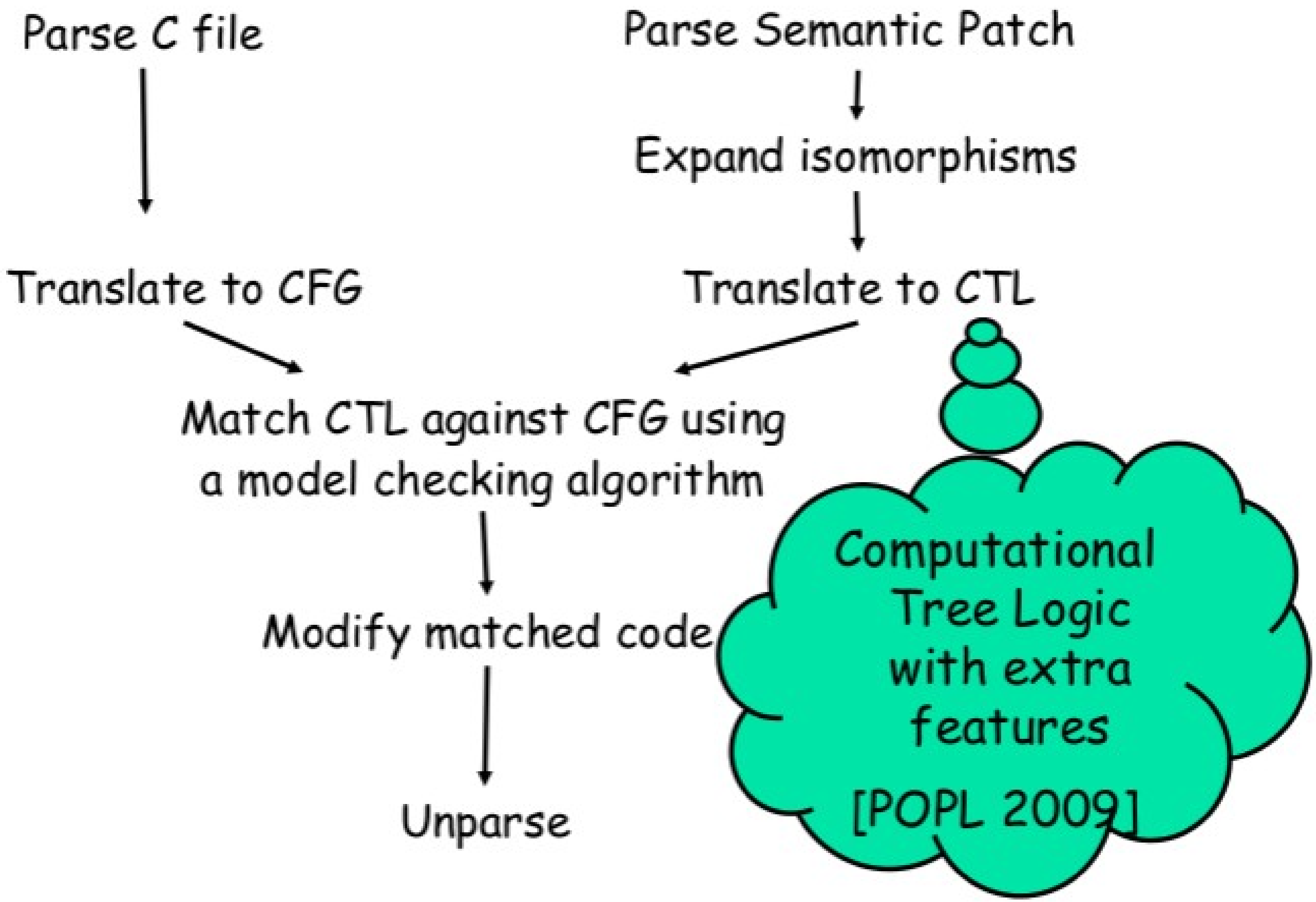
The Coccinelle tool

- Program matching and transformation for unpreprocessed C code.
- Scripts that can run every time we make a change to the file to ensure that the specific bugs are not being introduced.
- A single small semantic patch can modify hundreds of files, at thousands of code sites.
- Semantic Patch Language (SmPL):
  - Based on the syntax of patches
  - “Semantic Patch” notation abstracts and generalises “patches”.
  - Declarative approach to transformation
  - High level search that abstracts away from irrelevant details

Using SmPL to abstract away from irrelevant details

- Differences in spacing, indentation, and comments
- Give names to variables that can be expressions, statements, constants etc.
  - use of metavariables
- Irrelevant code
  - use of '...' operator
- Other variations in coding style (use of isomorphisms).
  - e.g. if(!y) <=> if(y==NULL) <=> if(NULL==y)
- Patch-like notation (−/+) for expressing transformations.

How does the Coccinelle work?



Example 1: Finding and fixing !x&y bugs

- The problem:
  - Combining a boolean (0/1) with a constant using & is usually meaningless.
  - In particular, if the rightmost bit of y is 0, the result will always be 0.

- Example:

```
                /* Did this counter overflow? */
-               if (!pmu_read_register(idx, CCI_PMU_OVRFLW) & CCI_PMU_OVRFLW_FLAG)
+               if (!(pmu_read_register(idx, CCI_PMU_OVRFLW) &
+                       CCI_PMU_OVRFLW_FLAG))
                        continue;
```

- The solution: Add parentheses.

The semantic patch

```
@@ expression E; constant C; @@
(
  !E & !C
|
- !E & C
+ !(E & C)
)
```

- Here, y is a constant.
- We have a disjunction so that no transformation takes place when y is itself negated, as an expression of the form !x&!y may make sense.

Example 2: Inconsistent API usage

Do we need this function?

```
static inline dma_addr_t
pci_map_single(struct pci_dev *hwdev, void *ptr, size_t size,
               int direction)
{
  return dma_map_single(hwdev == NULL ? NULL : &hwdev->dev, ptr,
                        size, (enum dma_data_direction)direction);
}
```

The use of pci_map_single

```
dev->phys_dma_addr =
    pci_map_single(dev->pci_dev, (void *)buf, R852_DMA_LEN,
      (do_read ? PCI_DMA_FROMDEVICE : PCI_DMA_TODEVICE));
```

would be more uniform as:

```
dev->phys_dma_addr =
    dma_map_single(&dev->pci_dev->dev, (void *)buf, R852_DMA_LEN,
      (do_read ? DMA_FROM_DEVICE : DMA_TO_DEVICE));
```

PCI constants

```
/* This defines the direction arg
 to the DMA mapping routines. */
#define PCI_DMA_BIDIRECTIONAL   0
#define PCI_DMA_TODEVICE        1
#define PCI_DMA_FROMDEVICE      2
#define PCI_DMA_NONE            3
```

DMA constants

```
enum dma_data_direction {
        DMA_BIDIRECTIONAL = 0,
        DMA_TO_DEVICE = 1,
        DMA_FROM_DEVICE = 2,
        DMA_NONE = 3,
};
```

The semantic patch

```
@@ expression E1,E2,E3,E4; @@
- pci_map_single(E1,
+ dma_map_single(&E1->dev,
     E2, E3, E4)

@@ expression E1,E2,E3; @@
dma_map_single(E1, E2, E3,
(
-    PCI_DMA_BIDIRECTIONAL
+    DMA_BIDIRECTIONAL
|
-    PCI_DMA_TODEVICE
+    DMA_TO_DEVICE
|
-    PCI_DMA_FROMDEVICE
+    DMA_FROM_DEVICE
|
-    PCI_DMA_NONE
+    DMA_NONE_DEVICE
)
  )
```

- Change function name.
- Add field access to the first argument.
- Rename the fourth argument.

Example 3: Dereference of a possibly NULL value

```
- struct sock *sk = tun->sk;
+ struct sock *sk;
  unsigned int mask = 0;

  if (!tun)
    return POLLERR;

+ sk = tun->sk;
```

Here, tun was being dereferenced before a NULL test.

The semantic patch

```
@@
type T;
expression E;
identifier i,fld,f1;
statement S;
@@

- T i = E->fld;
+ T i;
  ... when != E
      when != i
      when != f1(...,&E,...)
  if (E == NULL) S
+ i = E->fld;
```

- Find cases where a pointer is dereferenced and then compared with NULL.
- A very special case where the dereference is part of a declaration.
- Isomorphisms cause

  E == NULL to also match eg !E.

Example 4: Devm functions

- There are managed interfaces for allocating resources. Example: devm_kzalloc, devm_ioremap etc.

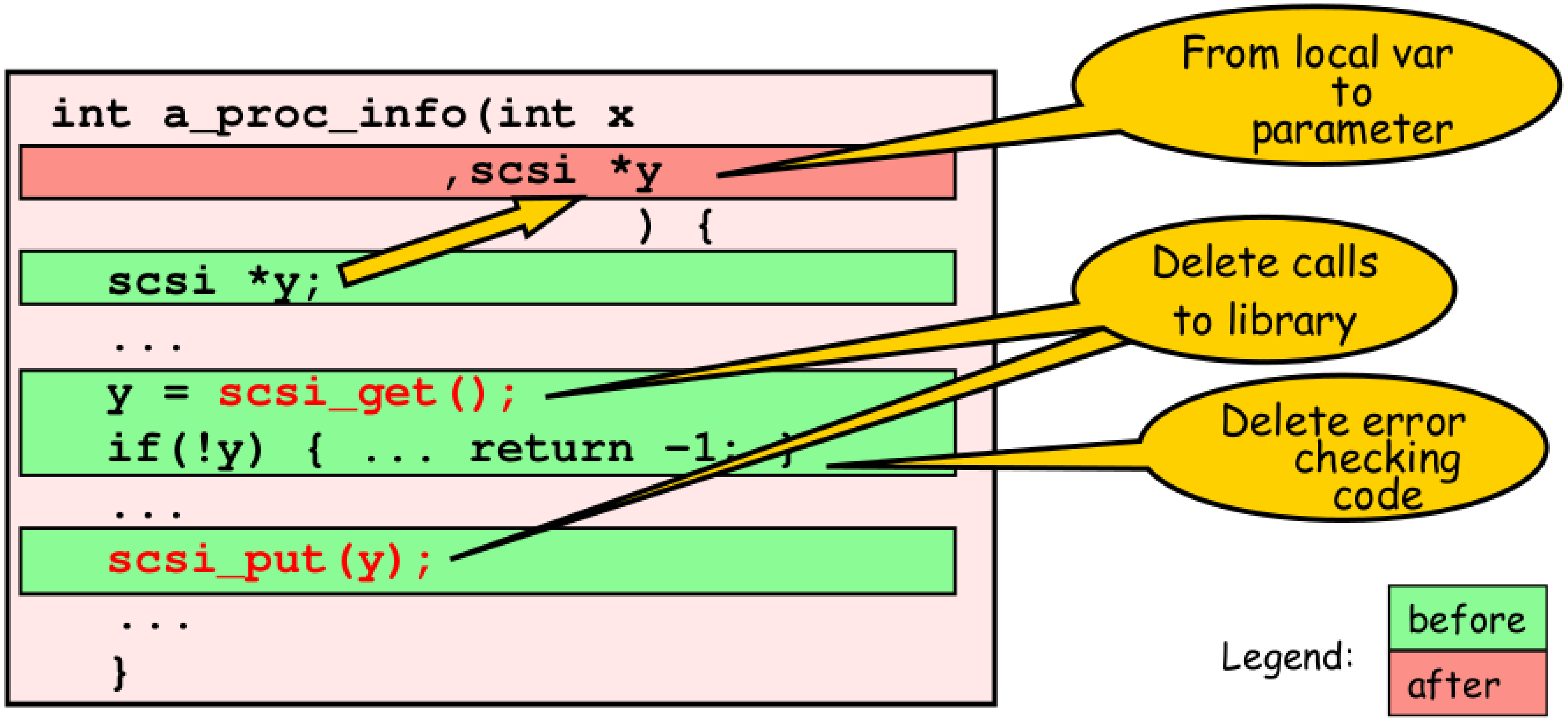
```
@platform@
identifier p, probefn, removefn;
@@
struct platform_driver p = {
  .probe = probefn,
  .remove = removefn,
};

@prb@
identifier platform.probefn, pdev;
expression e, e1, e2;
@@
probefn(struct platform_device *pdev, ...) {
  <+...
- e = kzalloc(e1, e2)
+ e = devm_kzalloc(&pdev->dev, e1, e2)
  ...
?-kfree(e);
  ...+>
}
```

```
@rem depends on prb@
identifier platform.removefn;
expression e;
@@
removefn(...) {
  <...
- kfree(e);
  ...>
}
```

- Convert kzalloc to devm_kzalloc.
- Kfrees are no longer required in the probe and remove functions.

Example 5: Remove get and put

- Evolution: scsi_get()/scsi_put() dropped from SCSI library.
- Collateral evolutions: SCSI resource now passed directly to proc_info callback functions via a new parameter.



Semantic patch

proc_info.sp

```
@@
function a_proc_info;
identifier x,y;
@@
 int a_proc_info(int x
+                    ,scsi *y
                        ) {
-   scsi *y;
    ...
-   y = scsi_get();
-   if(!y) { ... return -1; }
    ...
-   scsi_put(y);
      ...
  }
```

```
$ spatch -sp_file proc_info.sp
  -dir linux-next
```

```
int s53c700_info(int limit)
{
  char *buf;
  scsi *sc;
  sc = scsi_get();
  if(!sc) {
     printk("error");
     return -1;
  }
  wd7000_setup(sc);
  PRINTP("val=%d",
            sc->field+limit);
  scsi_put(sc);
  return 0;
}
```

```
int s53c700_info(int limit, scsi *sc)
{
  char *buf;







  wd7000_setup(sc);
  PRINTP("val=%d",
            sc->field+limit);

  return 0;
}
```

/linux/scripts/coccinelle!!

```
virtual patch
virtual context
virtual org
virtual report

//----------------------------------------------------
//  For context mode
//----------------------------------------------------

@depends on context@
expression e;
@@

*if (e) BUG();

//----------------------------------------------------
//  For patch mode
//----------------------------------------------------

@depends on patch@
expression e;
@@

-if (e) BUG();
+BUG_ON(e);
```

```
//----------------------------------------------------
//  For org and report mode
//----------------------------------------------------

@r@
expression e;
position p;
@@

 if (e) BUG@p ();

@script:python depends on org@
p << r.p;
@@

coccilib.org.print_todo(p[0], "WARNING use BUG_ON")

@script:python depends on report@
p << r.p;
@@

msg="WARNING: Use BUG_ON"
coccilib.report.print_report(p[0], msg)
```

Things to remember while using Coccinelle

- The semantic patches can have multiple rules.
- The rules are applied file by file in the same order as they appear in the semantic patch.
- We can have * in the patch to only find patterns but not transform anything.(context mode)
- Positions can be marked and relevant information such as line number and the variable names can be printed as messages. (report and org modes)
- To check if the syntax of the script is right, run:

```
spatch --parse-cocci sp.cocci
```

Nothing is perfect.

- Including header files increases running time:

  --no-includes --include-headers

- Pretty printing.
- Warnings or error messages are not very informative.


```
init_defs_builtins: /usr/local/share/coccinelle/standard.h
46 60
Fatal error: exception Failure("plus: parse error:
 = File "iserrnull.cocci", line 6, column 2,  charpos = 46
    around = 'IS_ERR_OR_NULL', whole content = + IS_ERR_OR_NULL(e)
")
```

Conclusion

- A patch-like program matching and transformation language
- Over 450 patches created using Coccinelle are being used to develop the Linux kernel. (Coccinellery)
- 49 patches in the Linux kernel itself, and a makefile target (make coccicheck) for running them, on the whole kernel, a particular subdirectory, or files with uncommitted changes.
- Looks like a patch; fits with Systems (Linux) programmers' habits.
- Quite "easy" to learn; widely accepted by the Linux community.
- Probable bugs found in gcc, postgresql, vim, amsn, pidgin, mplayer, openssl, vlc, wine.

Thank you

Himangi Saraogi

himangi774@gmail.com

Website: http://web.iiit.ac.in/~himangi.saraogi

http://himangi99.wordpress.com/