Introduction to Coccinelle

Julia Lawall (Inria/LIP6)

http://coccinelle.lip6.fr

October 31, 2013

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

Our goals

- Automatically find code containing bugs or defects, or requiring collateral evolutions.
- Automatically fix bugs or defects, and perform collateral evolutions.
- Provide a system that is accessible to software developers.

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.

Coccinelle

Program matching and transformation for unpreprocessed C code.

Fits with the existing habits of C programmers.

- C-like, patch-like notation

Semantic patch language (SmPL):

- Metavariables for abstracting over subterms.
- "..." for abstracting over code sequences.
- Patch-like notation (−/+) for expressing transformations.

The !& problem

The problem: Combining a boolean (0/1) with a constant using & is usually meaningless:

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

The solution: Add parentheses.

Our goal: Do so automatically for any expression E and constant C.

A semantic patch for the !& problem

```
@@
expression E;
constant C;
@@

- !E & C
+ !(E & C)
```

Two parts per rule:

- Metavariable declaration
- Transformation specification

A semantic patch can contain multiple rules.

Metavariable types

Surrounded by `@@ @@`.

- expression, statement, type, constant, local idexpression
- A type from the source program
- iterator, declarer, iterator name, declarer name, typedef

Transformation specification

- − in the leftmost column for something to remove
- + in the leftmost column for something to add
- * in the leftmost column for something of interest
  - Cannot be used with + and −.
- Spaces, newlines irrelevant.

Exercise 1

1. Create a file ex1.cocci containing the following:

   ```
   @@
   expression E;
   constant C;
   @@

   - !E & C
   + !(E & C)
   ```

2. Run spatch: `spatch --sp-file ex1.cocci --dir linux-3.2/drivers/staging/crystalhd`

3. Did your semantic patch do everything it should have?

4. Did it do something it should not have?

Exercise 2

Some code contains a cast on the result of kmalloc. For example:

```
info->RegsBuf = (unsigned char *)
  kmalloc(sizeof(info->ATARegs), GFP_KERNEL);
```

If the destination of the returned value has pointer type, this cast is not needed.

1. Complete the following semantic patch to remove this unnecessary cast.

   ```
   @@ expression * e; expression arg1, arg2; type T; @@

   [fill it in]
   ```

2. Test your semantic patch on the code in `linux-3.2/drivers/isdn`
3. Are you satisfied with the appearance of the results? If not, try to improve it.

Practical issues

To check that your semantic patch is valid:

```
spatch --parse-cocci mysp.cocci
```

To run your semantic patch:

```
spatch --sp-file mysp.cocci file.c
spatch --sp-file mysp.cocci --dir directory
```

To understand why your semantic patch didn't work:

```
spatch --sp-file mysp.cocci file.c --debug
```

If you don't need to include header files:

```
spatch --sp-file mysp.cocci --dir directory
                 --no-includes --include-headers
```

More practical issues

Put the interesting output in a file:

```
spatch ... > output.patch
```

Omit the uninteresting output:

```
spatch --very-quiet ...
```

The source code:

```
/usr/src/linux-source-3.2/scripts/coccinelle/
```

These slides:

```
http://pagesperso-systeme.lip6.fr/Julia.Lawall/tutorial/
part1.pdf
```

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

The code:

```
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));
```

Issues:

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

pci_map_single: Example and definitions

Commit b0eb57cb

```
- rbi->dma_addr = pci_map_single(adapter->pdev,
+ rbi->dma_addr = dma_map_single(
+    &adapter->pdev->dev,
     rbi->skb->data, rbi->len,
     PCI_DMA_FROMDEVICE);
```

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,
};
```

pci_map_single: First attempt

Outline of a semantic patch, including the patch example:

```
@@

@@

- rbi->dma_addr = pci_map_single(adapter->pdev,
+ rbi->dma_addr = dma_map_single(
+     &adapter->pdev->dev,
      rbi->skb->data, rbi->len,
      PCI_DMA_FROMDEVICE);
```

pci_map_single: First attempt

Eliminate irrelevant code:

```
@@

@@

- pci_map_single(adapter->pdev,
+ dma_map_single(
+    &adapter->pdev->dev,
     rbi->skb->data, rbi->len,
     PCI_DMA_FROMDEVICE)
```

pci_map_single: First attempt

Abstract over subterms:

```
@@
expression E1,E2,E3;
@@

- pci_map_single(E1,
+ dma_map_single(
+     &E1->dev,
      E2, E3,
      PCI_DMA_FROMDEVICE)
```

pci_map_single: First attempt

Rename the fourth argument:

```
@@
expression E1,E2,E3;
@@

- pci_map_single(E1,
+ dma_map_single(
+     &E1->dev,
      E2, E3,
-     PCI_DMA_FROMDEVICE)
+     DMA_FROM_DEVICE)
```

pci_map_single: Second attempt

Need to consider all direction constants.

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

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

Etc. Four rules in all.

pci_map_single: Third attempt

Avoid code duplication: Use a disjunction.

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

pci_map_single: Fourth attempt

```
@@ 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
)
  )
```

Exercise 3

1. Implement some version of the semantic patch for converting calls to pci_map_single to calls to dma_map_single.
2. Test your implementation on the directory linux-3.2/drivers/net/ethernet.
3. Implement both the third version and the fourth version. Compare the results.
4. Other PCI functions replicate DMA behavior, *e.g.*, pci_unmap_single. For example, commit b0eb57cb contains:

   ```
   -   pci_unmap_single(pdev, tbi->dma_addr, tbi->len,
   +   dma_unmap_single(&pdev->dev, tbi->dma_addr, tbi->len,
                                        PCI_DMA_TODEVICE);
   ```

   Extend your semantic patch to implement this transformation. Try to minimize the number of rules.

Getter and setter functions

Some functions from `include/linux/ide.h`:

```
static inline void *
ide_get_hwifdata (ide_hwif_t * hwif)
{
        return hwif->hwif_data;
}

static inline void
ide_set_hwifdata (ide_hwif_t * hwif, void *data)
{
        hwif->hwif_data = data;
}
```

Goal: Replace uses of `hwif->hwif_data` by calls to these function.

Getter and setter functions: First attempt

```
@@
expression hwif;
@@
- hwif->hwif_data
+ ide_get_hwifdata(hwif)

@@
expression hwif, data;
@@
- hwif->hwif_data = data
+ ide_set_hwifdata(hwif, data)
```

Problems

```
@@ expression hwif; @@
- hwif->hwif_data
+ ide_get_hwifdata(hwif)
```

- The rule applies to

  ```
  unsigned long base = (unsigned long)hwif->hwif_data;
  ```

  but also to

  ```
  hwif->hwif_data = NULL;
  ```

- ide_get_hwifdata has prototype:

  ```
  static inline void *ide_get_hwifdata (ide_hwif_t * hwif);
  ```

  The rule transforms all hwif_data field references.

Second attempt: Rule order

```
@@
expression hwif, data;
@@
- hwif->hwif_data = data
+ ide_set_hwifdata(hwif, data)

@@
expression hwif;
@@
- hwif->hwif_data
+ ide_get_hwifdata(hwif)
```

Applies to 9 code sites, in 2 files.

Third attempt: Metavariable type constraints

```
@@ ide_hwif_t *hwif; expression data; @@
- hwif->hwif_data = data
+ ide_set_hwifdata(hwif, data)

@@ ide_hwif_t *hwif; @@
- hwif->hwif_data
+ ide_get_hwifdata(hwif)
```

Can optionally add `typedef ide_hwif_t;` in the first rule.

- Typedef is needed when the type appears only in a cast.
- Typedef appears only once, where earliest needed.

Exercise 4

1. Implement all three variants of the semantic patch for introducing the `ide_get_hwifdata` and `ide_set_hwifdata` getter and setter functions.
2. Test each variant on the directory `linux-3.2/drivers/ide`, and compare the results.
3. Reimplement each variant using a disjunction.
4. Compare the results of the disjunction variants to the original implementations.

Exercise 5

In the case of `pci_map_single` and `dma_map_single`, we could also prefer to convert PCI occurrences of `dma_map_single` to calls to `pci_map_single` (i.e., the reverse transformation).

1. Implement a semantic patch to do this transformation.
2. Test your semantic patch on the directory `linux-3.2/drivers/net/ethernet`.
3. Given the definition of `pci_map_single` in `include/asm-generic/pci-dma-compat.h`, why should the file `ethernet/cadence/macb.c` not be transformed?
4. Check that your semantic patch makes no modification in this file.

Summary

SmPL features seen so far:

- Metavariables for abstracting over arbitrary expressions.
- Metavariables restricted to particular types.
- Disjunctions.
- Multiple rules.
- Rule ordering.