LF *live*

MENTORSHIP SERIES

LF Live MENTORSHIP SERIES

# Coccinelle: Automating Large-scale Evolution and Bug Finding in C Code

Julia Lawall, Senior Researcher, Inria

# Common programming problems

- Programmers don’t always think about how C works.
  - if (unlikely(aligned.ptr - source->ptr) > 0) doesn’t make sense.
- 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.

LF Live MENTORSHIP SERIES

# Example: badly placed unlikely

```
diff --git a/drivers/platform/surface/aggregator/ssh_packet_layer.c
b/drivers/platform/surface/aggregator/ssh_packet_layer.c
@@ -1694,7 +1694,24 @@ static size_t ssh_ptl_rx_eval(struct ssh_ptl *ptl, struct ssam_span
*source)
        /* Find SYN. */
        syn_found = sshp_find_syn(source, &aligned);
-       if (unlikely(aligned.ptr - source->ptr) > 0) {
+       if (unlikely(aligned.ptr != source->ptr)) {
```

# Example: Inconsistent API usage

drivers/char/xillybus/xillybus_pcie.c

```
addr = pci_map_single(ep->pdev, ptr, size, pci_direction);
if (pci_dma_mapping_error(ep->pdev, addr)) {
        kfree(this);
        return -ENODEV;
}
```

drivers/media/pci/solo6x10/solo6x10-p2m.c

```
dma_addr = dma_map_single(&solo_dev->pdev->dev, sys_addr, size,
                          wr ? DMA_TO_DEVICE : DMA_FROM_DEVICE);
if (dma_mapping_error(&solo_dev->pdev->dev, dma_addr))
        return -ENOMEM;
```

Why not pci functions in all cases? Or dma functions in all cases?

# Example: Missing error check

kzalloc returns NULL on insufficient memory.

```
si_data = kzalloc(data_size, __GFP_DMA | GFP_KERNEL);
cbd.length = cpu_to_le16(data_size);
dma = dma_map_single(&priv->si->pdev->dev, si_data, data_size, DMA_FROM_DEVICE);
if (dma_mapping_error(&priv->si->pdev->dev, dma)) {
        ...
        return -ENOMEM;
}
...
eth_broadcast_addr(si_data->dmac);
```

From drivers/net/ethernet/freescale/enetc/enetc_qos.c

LF live MENTORSHIP SERIES

# Our goals

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

# Requirements for automation

The ability to abstract over irrelevant information:

- if (unlikely(aligned.ptr - source->ptr) > 0): aligned.ptr is not important.

The ability to match scattered code fragments:

- kzalloc call may be far from the first dereference.

The ability to transform code fragments:

- Replace pci_map_single by dma_map_single, or vice versa.

LF Live MENTORSHIP SERIES

# Coccinelle to the rescue!

# 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.

LF Live MENTORSHIP SERIES

# Simple SmPL: A semantic patch for the “unlikely” problem

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

- unlikely(E1) > E2
+ unlikely(E1 > E2)
```

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

LF Live MENTORSHIP SERIES

# 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.

LF Live MENTORSHIP SERIES

# Exercise 1

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

   ```
   @@
   expression E1, E2;
   @@
   -  unlikely(E1) != E2
   + unlikely(E1 != E2)
   ```

2. Run spatch: spatch --iso-file empty.iso --sp-file ex1.cocci --dir linux/arch/x86/events

   or: spatch --iso-file empty.iso --sp-file ex1.cocci --dir linux/drivers/net/ethernet/freescale

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

4. Did it do something it should not have?

LF Live MENTORSHIP SERIES

# Exercise 2

1. A commonly useful change is to use a specific setter function, as follows:

```
-   dev_set_drvdata(&pdev->dev, pmic);
+   platform_set_drvdata(pdev, pmic);
```

   Complete the following semantic patch (ex2.cocci) to make this change more generally:

```
@@ expression E1, E2; @@
[ fill in this part ]
```

2. Run spatch: spatch --sp-file ex2.cocci --dir linux/drivers/mfd
3. Try to compile each affected file, eg
   make allyesconfig drivers/mfd/intel_soc_pmic_bxtwc.o
4. Did the semantic patch do something it should not have?

# 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 …
```

Concurrency:

```
spatch -j 4 ...
```

# 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);
}
```

LF live MENTORSHIP SERIES

# The use of pci map single

The code:

```
dma_addr = pci_map_single(nic->pdev, skb->data, skb->len, PCI_DMA_TODEVICE);
```

would be more uniform as:

```
dma_addr = dma_map_single(&nic->pdev->dev, skb->data, skb->len, DMA_TO_DEVICE);
```

Issues:

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

LF Live MENTORSHIP SERIES

# 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:

```
@@


@@
- 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:

```
@@
expression E1, E2, E3;
@@
- 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: 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)
```

- Fixes 17 calls
- 43 calls remain...

# 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, E2, E3
+ 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
)
  )
```

# 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
)
    )
```

LF Live MENTORSHIP SERIES

# 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 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, PCI_DMA_TODEVICE);
+       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.

LF live MENTORSHIP SERIES

# Dots

## Issue:

- Sometimes it is necessary to search for multiple related code fragments.

## Goals:

- Specify patterns consisting of fragments of code connected by execution paths.
- Specify constraints on the contents of those execution paths.

# Example: Inadequate error checking of kzalloc

kzalloc returns NULL on insufficient memory.

Good code:

```
check_state = kzalloc(sizeof(struct btree_check_state), GFP_KERNEL);
if (!check_state)
        return -ENOMEM;
```

Bad code:

```
ddip = kzalloc(sizeof(struct detached_dev_io_private), GFP_NOIO);
ddip->d = d;
```

# More bad code

```
si_data = kzalloc(data_size, __GFP_DMA | GFP_KERNEL);
cbd.length = cpu_to_le16(data_size);
dma = dma_map_single(&priv->si->pdev->dev, si_data, data_size, DMA_FROM_DEVICE);
if (dma_mapping_error(&priv->si->pdev->dev, dma)) {
        ...
        return -ENOMEM;
}
...
eth_broadcast_addr(si_data->dmac);
```

The kmalloc and the dereference are not necessarily contiguous.

# Using dots

Start with a typical example of code

```
si_data = kzalloc(data_size, __GFP_DMA | GFP_KERNEL);
cbd.length = cpu_to_le16(data_size);
dma = dma_map_single(&priv->si->pdev->dev, si_data, data_size, DMA_FROM_DEVICE);
if (dma_mapping_error(&priv->si->pdev->dev, dma)) {
        ...
        return -ENOMEM;
}
...
eth_broadcast_addr(si_data->dmac);
```

LF Live MENTORSHIP SERIES

# Using dots

## Highlight what is wanted

```
        si_data = kzalloc(data_size, __GFP_DMA | GFP_KERNEL);
        cbd.length = cpu_to_le16(data_size);
        dma = dma_map_single(&priv->si->pdev->dev, si_data, data_size, DMA_FROM_DEVICE);
        if (dma_mapping_error(&priv->si->pdev->dev, dma)) {
                ...
                return -ENOMEM;
        }
        ...
*       eth_broadcast_addr(si_data->dmac);
```

# Using dots

Replace the irrelevant statements by ...

```
si_data = kzalloc(data_size, __GFP_DMA | GFP_KERNEL);
...
```

```
*       eth_broadcast_addr(si_data->dmac);
```

# Using dots

Abstract over irrelevant subterms.

- May use …

```
@@
expression e; identifier f;
@@
e = kzalloc(...);
…
* e->f
```

# Some tests

Matches:

```
        ddip = kzalloc(sizeof(struct detached_dev_io_private), GFP_NOIO);
-       ddip->d = d;
```

# Some tests

Matches:

```
        ddip = kzalloc(sizeof(struct detached_dev_io_private), GFP_NOIO);
-       ddip->d = d;
```

Also matches:

```
        check_state = kzalloc(sizeof(struct btree_check_state), GFP_KERNEL);
        if (!check_state)
                return -ENOMEM;
-       check_state->c = c;
```

# Some tests

Matches:

```
        ddip = kzalloc(sizeof(struct detached_dev_io_private), GFP_NOIO);
-       ddip->d = d;
```

Also matches:

```
        check_state = kzalloc(sizeof(struct btree_check_state), GFP_KERNEL);
        if (!check_state)
                return -ENOMEM;
-       check_state->c = c;
```

Need to stop on reaching !check_state.

# Using dots

## Stopping at NULL tests.

```
@@
expression e; identifier f;
@@
e = kzalloc(...);
…
(
e == NULL
|
e != NULL
|
*e->f
)
```

# Some more tests

Matches:

```
        ddip = kzalloc(sizeof(struct detached_dev_io_private), GFP_NOIO);
-       ddip->d = d;
```

Does not match:

```
        check_state = kzalloc(sizeof(struct btree_check_state), GFP_KERNEL);
        if (!check_state)
                return -ENOMEM;
        check_state->c = c;
```

# Another problem

Also matches:

```
        ia = kzalloc(sizeof(*ia), GFP_KERNEL | __GFP_NOFAIL);
-       ia->in.major = FUSE_KERNEL_VERSION;
```

# Avoiding __GFP_NOFAIL

```
@@ expression e, x; identifier f; @@
(
e = kzalloc(...,x | __GFP_NOFAIL);
|
e = kzalloc(...);
…
(e == NULL
|e != NULL
|
*e->f
)
)
```

# Avoiding __GFP_NOFAIL

```
@@ expression e, x; identifier f; @@
(
e = kzalloc(...,x | __GFP_NOFAIL);
|
e = kzalloc(...);
…
(e == NULL
|e != NULL
|
*e->f
)
)
```

30 results.

16 bugs, 14 false positives

- Aliases
- Alternate tests
- Evaluation of the results takes a few minutes.

# A final improvement

```
@@ expression e, e1, x; identifier f; @@
(
e = kzalloc(...,x | __GFP_NOFAIL);
|
e = kzalloc(...);
… when != e1 = e
    when != e = e1
(e == NULL
|e != NULL
|
*e->f
)
)
```

26 results.

16 bugs, 10 false positives

- Aliases
- Alternate tests
- Evaluation of the results takes a few minutes.

LF Live MENTORSHIP SERIES

# Exercise 4

The result of dma_map_single should be tested with dma_mapping_error:

```
dma_addr = dma_map_single(&solo_dev->pdev->dev, sys_addr, size,
                          wr ? DMA_TO_DEVICE : DMA_FROM_DEVICE);
if (dma_mapping_error(&solo_dev->pdev->dev, dma_addr))
        return -ENOMEM;
```

1. Write a semantic patch to verify that this is done.
2. Run your semantic patch on drivers/scsi
3. Do some results appear to be false positives? How could you improve your semantic patch?

# Exercise 5

One of the results for the kmalloc with no NULL test example is (drivers/macintosh/smu.c):

```
        pp = kzalloc(sizeof(struct smu_private), GFP_KERNEL);
        if (pp == 0)
                return -ENOMEM;
-       spin_lock_init(&pp->lock);
```

The code will not crash, but it is not as nice as it could be. Write a semantic patch to replace such uses of 0 by NULL.

Hints:

- A metavariable of type “expression *” matches any pointer typed expression.
- This exercise has nothing to do with dots.

# Summary

SmPL features seen so far:

- Metavariables for abstracting over arbitrary expressions.
- Disjunctions.
- Multiple rules.
- Dots and when.

LF live MENTORSHIP SERIES

# Advanced SmPL

## Isomorphisms:

- Write x == NULL. Match !x.

## Typed metavariables

## Rule names and rule ordering:

- Later rules see the results of earlier rules.

## Python, OCaml interface:

- Print warning messages.
- Position metavariables give access to position information.

# Isomorphisms

## Issue:

- Coccinelle matches code exactly as it appears.
- x == NULL does not match !x.

## Goal:

- Transparently treat similar code patterns in a similar way.

LF Live MENTORSHIP SERIES

# Isomorphisms

An isomorphism relates code patterns that are considered to be similar:

Expression
@ is_null @ expression X; @@
X == NULL <=> NULL == X => !X

Expression
@ paren @ expression E; @@
(E) => E

Implemented by semantic patch rewriting.

# Example

```
@@
expression n, d;
@@
-  (((n) + (d) - 1) / (d))
+ DIV_ROUND_UP(n,d)
```

Changes 287 occurrences in Linux 5.11-rc5.

# Typed metavariables

```
@@
expression pdev, e;
@@
-       dev_set_drvdata(&pdev->dev, e);
+       platform_set_drvdata(pdev, e);
```

# Typed metavariables

```
@@
expression pdev, e;
@@
-       dev_set_drvdata(&pdev->dev, e);
+       platform_set_drvdata(pdev, e);
```

Actually, only valid if pdev has type struct platform_device *.

Other types require other accessor functions.

# Typed metavariables

```
@@
struct platform_device * pdev; expression e;
@@
-       dev_set_drvdata(&pdev->dev, e);
+       platform_set_drvdata(pdev, e);
```

Actually, only valid if pdev has type struct platform_device *.

Other types require other accessor functions.

LF live MENTORSHIP SERIES

# Rule names and rule ordering

Goal: Find probe functions that call kmalloc (possible devm_kmalloc candidates)

1. Find a probe function (function stored in a probe field).
2. Check whether the function calls kmalloc.

Problem: Need to match a structure and then a function definition,
but a SmPL rule matches only one thing at a time.

# Rule names and rule ordering

```
@r@ identifier i, j, fn; @@
struct i j = { .probe = fn, };

@@ identifier r.fn; @@
fn(...) {
  <...
* kmalloc(...)
  ...>
}
```

Finds matches in 90 files.

# Positions and Python

```
@r@ identifier i, j, fn; @@
struct i j = { .probe = fn, };

@@ identifier r.fn; @@
fn(...) {
  <...
* kmalloc(...)
  ...>
}
```

- * is useful for emacs diff mode
- In other settings, may prefer a helpful message.

# Positions and Python

```
@r@ identifier i, j, fn; @@
struct i j = { .probe = fn, };

@s@ identifier r.fn; position p; @@
fn(...) {
  <...
  kmalloc@p(...)
  ...>
}
```

```
@script:python@
fn << r.fn;
p << s.p;
@@
file = p[0].file
line = p[0].line
print “%s:%s: kmalloc in probe function %s” %
        (file,line,fn)
```

LF live MENTORSHIP SERIES

# Conclusion

- Coccinelle provides a declarative language for program matching and transformation.
- Coccinelle semantic patches look like patches; fit with Linux developers' habits.
- Quite "easy" to learn; accepted by the Linux community.
- Some other tools building on Coccinelle:
  - Spinfer: semantic patch inference from examples
  - Prequel: patch queries for searching git histories
  - PatchNet: neural network for identifying bug fixing patches

https://coccinelle.gitlabpages.inria.fr/website/

LF live MENTORSHIP SERIES

# Thank you for joining us today!

We hope it will be helpful in your journey to learning more about effective and productive participation in open source projects. We will leave you with a few additional resources for your continued learning:

- The LF Mentoring Program is designed to help new developers with necessary skills and resources to experiment, learn and contribute effectively to open source communities.
- Outreachy remote internships program supports diversity in open source and free software
- Linux Foundation Training offers a wide range of free courses, webinars, tutorials and publications to help you explore the open source technology landscape.
- Linux Foundation Events also provide educational content across a range of skill levels and topics, as well as the chance to meet others in the community, to collaborate, exchange ideas, expand job opportunities and more. You can find all events at events.linuxfoundation.org.