INTRODUCTION TO COCCINELLE AND SMPL

[Solutions of exercises]

Linuxcon Japan, 2016

Vaishali Thakkar (vaishali.thakkar@oracle.com)

Exercise 1

- Semantic patch 1:

```
@@
constant c;
@@

-1 << c
+BIT(c)
```

- Semantic patch 2:

```
@@
expression E;
@@

-1 << E
+BIT(E)
```

Exercise 2

- Semantic patch 1:

[Removing paranthesis for the bitwise left shift operations]

```
@@
identifier i;
constant c;
type t;
expression e;
@@

t i =
-(e
+e
<<
-c);
+c;
```

Exercise 2

- Semantic patch 2:

[Removing paranthesis around the function arguments]

```
@@
expression e, e1;
identifier f;
constant c;
@@

e1 = f(...,
-(e
+e
<<
-c)
+c
,...);
```

Exercise 3

- Semantic patch:

  [Good example of isomorphism]

```
@@
expression n,d;
@@

- (((n) + (d)) - 1) / (d))
+ DIV_ROUND_UP(n,d)
```

Exercise 4

- Semantic patch:

[Good example of isomorphism]

```
@ haskernel @
@@

#include <linux/kernel.h>

@ depends on haskernel @
expression n,d;
@@

- (((n) + (d)) - 1) / (d))
+ DIV_ROUND_UP(n,d)
```

Exercise 5

- Semantic patch:

[Semantic patch handling both cases]

```
@match_immediate_function_data_after_init_timer@
expression e, func, da;
@@

-init_timer (&e);
+setup_timer (&e, func, da);

(
-e.function = func;
-e.data = da;
|
-e.data = da;
-e.function = func;
)
```

Exercise 6

- Semantic patch 1:

[Use of dots for the setup_timer cases]

```
@match_function_and_data_after_init_timer@
expression e, e1, e2, e3, e4, func, da;
@@

-init_timer (&e);
+setup_timer (&e, func, da);

...

(
-e.function = func;
...
-e.data = b;
|
-e.data = da;
...
-e.function = a;
)
```

Exercise 6

- Semantic patch 2:

[Semantic patch rule when data is not initialized]

```
@match_function_after_init_timer@
expression e5, e6, fun;
@@

-init_timer (&e5);
+setup_timer (&e5, fun, 0UL);
...
-e6.function = fun;
```

Exercise 7

- Semantic patch:

[Semantic patch with removing variable for compression of lines]

```
@@
expression ret;
identifier f;
@@

- T ret;
...
-ret =
+return
      f(...);
-return ret;
```

Exercise 8

- Semantic patch:

[Semantic patch to make sure that the variable is not used anywhere else]

```
@@
expression ret;
identifier f;
@@

- T ret;
... when != ret
-ret =
+return
     f(...);
-return ret;
```

Exercise 8

- Semantic patch:

[Semantic patch to include constants]

```
@@
type T;
constant C;
identifier ret;
@@
- T ret = C;
... when != ret
    when strict
return
- ret
+ C
;
```

Exercise 9

- Semantic patch:

```
@@
expression e;
type t;
identifier f;
@@

f(...,
-(t *)
e
,...)
```

Exercise 10

- Semantic patch:

```
@r@
expression e;
position p1, p2;
identifier f;
@@

e@p1 = f(...)
...
return e@p2;

@script:python@
p1 << r.p1; // inherit a metavariable p1 from rule r
p2 << r.p2; // inherit a metavariable p2 from rule r
@@
print p1[0].file, p1[0].line, p2[0].line
```

Exercise 11

- Semantic patch:

```
@@
expression ret;
identifier f;
@@

* T ret;
... when != ret
*ret =
*return ret;
```

[Check scripts/coccinelle/misc/returnvar.cocci for the reference.]

Exercise 12

- Semantic patch:

```
@r1 exists@
identifier f;
position p;
@@

f(...) { ... when any
  init_timer@p(...)
  ... when any
}
@r2 exists@
identifier g != r1.f;
struct timer_list t;
expression e8;
@@

g(...) { ... when any
  t.data = e8
  ... when any
}
@script:python depends on r2@
p << r1.p;
@@
cocci.include_match(False)
```

[Check scripts/coccinelle/api/setup_timer.cocci for the complete script.]