Code Transformation Evaluation

Contents

1 Code-imitator's code transformers **TRANSFORMER**

Contents

GitHub: `https://github.com/EQuiw/code-imitator`

Code-imitator's primary focus is on attack against authorship attribution of source code. Most of the code transformers do not include an `-all` flag, resulting in transformations that only affect a single instance of the transformation targets.

1.1 [95%] Compound statement transformations 0 **BAD**

Description

Adds a compound statement ({...}).

Executable: `compoundstmt_transformer`

Options:

- strategy: 2

Example:

```
// Before transformation
if (foo)
  bar();

// After transformation
if (foo) {
  bar();
 }
```

Total number of transformations in test/validation set: 693/669

Evaluation

Transformation	Good	Bad
`Compound_statement_transformations_0`	7558.c	9330.c
	24200.c	
	3499.c	
	3227.c	
	10253.c	
	10364.c	
	24880.c	
	23692.c	
	19724.c	
	20199.c	
	1672.c	
	21585.c	
	24720.c	
	23274.c	
	8340.c	
	12984.c	
	25130.c	
	12887.c	
	21925.c	

- No `-all` flag. Only applies the transformation to a single AST node.
- Illegal syntax produced by scoping the left side of a variable assignment. This applies to `9330.c`.

```
// test/9330.c
// Before transformation
rtype = sd_normal_command(sd, *req);

// After transforation
{ rtype } = sd_normal_command(sd, *req);
```

1.2 [100%] Compound statement transformations 1 GOOD

Description

Removes a compound statement ({. . .}).

Executable: `compoundstmt_transformer`

Options:

- strategy: 2

Example:

```
// Before transformation
if (foo) {
  bar();
}

// After transformation
if (foo)
  bar();
```

Does the opposite of the previous transformation, i.e., removes a compound statement.

Total number of transformations in test/validation set: 3/2

Evaluation

Transformation	Good	Bad
`Compound_statement_transformations_1`	3731.c	
	11113.c	
	17329.c	
	15262.c*	
	19866.c*	

Table 1: The star (*) symbol indicates that the transformation is from the validition set.

- No `-all` flag. Only applies the transformation to a single AST node.

1.3 [100%] Floating-point type transformations GOOD

Description

Converts floats to doubles.

Executable: `datastructure_transformer`

Options:

- strategy: 4

Example:

```
// Before transformation
float foo = 0.0;

// After transformation
double foo = 0.0;
```

Converts the type of a variable from `float` to `double`.

Total number of transformations in test/validation set: 17/23

Evaluation

Transformation	Good	Bad
`Floating-point_type_transformations`	5190.c	
	24787.c	
	3274.c	
	18095.c	
	19965.c	
	8085.c	
	13524.c	
	3971.c	
	2110.c	
	4260.c	
	5621.c	
	5044.c	
	60.c	
	7808.c	
	15249.c	
	22844.c	
	17943.c*	
	26459.c*	
	16464.c*	
	6517.c*	

Table 2: The star (*) symbol indicates that the transformation is from the validition set.

- No `-all` flag. Only applies the transformation to a single variable.

1.4 [55%] For statement transformations **BAD**

Description

Converts a for loop to a while loop.

Executable: `for_transformer`

Example:

```
// Before transformation
for (int i=0; i < 10; i++)
  foo();

// After transformation
{
  int i = 0;
  while (i < 10){
    foo();
    i++;
  }
}
```

Total number of transformations in test/validation set: 356/355

Evaluation

Transformation	Good	Bad
`For_statement_transformations`	35.c	12362.c
	5905.c	18254.c
	7716.c	19724.c
	18324.c	2182.c
	3293.c	25748.c
	22815.c	18279.c
	21427.c	24244.c
	26154.c	14654.c
	8955.c	11026.c
	23872.c	
	25472.c	

- No `-all` flag. Only applies the transformation to a single for loop.
- Produces invalid syntax: "extraneous ')' after condition, expected a statement". This applies to `12362.c`, `19724.c`, `2182.c`, `25748.c`, `18279.c`, `24244.c`, `14654.c`, `11026.c`.

```
// test/12362.c
// Before transformation
for(i=0; i<count; i++){ \* ... *\ }

// After transformation
{
  i=0;
  while (i<count)
    ){ // <- The extraneous ')'
  i++;
}}
```

- Missing conditional variable initialization. This applies to `18254.c`.

```
// test/18254.c
// Before transformation
for (i = CSR_RCVRL(s)-1; i > 0; i--, rcvrc--) { \* ... *\ }

// After transformation
```

```
while (i > 0)
{
    i--, rcvrc--;
}
```

- Introduces other syntactical errors. This applies to `2182.c`.

```
// test/2182.c
// Before transformation
for(i = nb_regs; i < nb_iargs; i++) {
  // ...
 }

// After transformation
        {
          i = nb_regs; i < n
            while (i < nb_iargs)
              {
                i++;    }
        }
```

1.5 [60%] Identifier transformations **BAD**

Description

Renames an identifier, i.e. a variable or a function.

Executable: `declname_transformer`

Example:

```
// Before transformation
unsigned int command;

// After transformation
unsigned int axdu_Var;
```

Total number of transformations in test/validation set: 2634/2655

Evaluation

Transformation	Good	Bad
`Identifier_transformations`	12946.c	21899.c
	10735.c	20831.c
	26552.c	13905.c
	21513.c	12558.c
	17439.c	14359.c
	1710.c	14723.c
	15582.c	8815.c
	2967.c	5965.c
	11142.c	
	11674.c	
	2950.c	
	20368.c	

- No `-all` flag. Only applies the transformation to a single identifier.
- Does not update to the new name in all usage instances. This applies to `21899.c`, `20831.c`, `13905.c`, `12558.c`, `14359.c`, `14723.c`, `8815.c`, `5965.c`.

1.6 [100%] Include typedef transformations **GOOD**

Description

Inserts a type using typedef, and updates all locations where the new type can be used. Defaults are extracted from the 2016 Code Jam Competition.

Executable: `typedef_transformer`

Options:

- strategy: `addtypedefs`
- selection: `selectrandomly`

Example:

```
// Before transformation
double sync_ipts;
double vdelta = sync_ipts - ost->sync_opts;

// After transformation
typedef double td_d; // Modification outside function scope
//..
td_d sync_ipts;
td_d vdelta = sync_ipts - ost->sync_opts;
```

Total number of transformations in test/validation set: 701/680

Evaluation

Transformation	Good	Bad
`Include-typedef_transformations`	20129.c	
	9095.c	
	12281.c	
	18512.c	
	6763.c	
	16812.c	
	16507.c	
	2656.c	
	25578.c	
	7069.c	
	21100.c	
	2395.c	
	12846.c	
	461.c	
	26396.c	
	27151.c	
	7360.c	
	5102.c	
	14306.c	
	17314.c	

- Adds code outside function scope.
- No `-all` flag. Sometimes applies the transformation to several variable declarations, other times only to a single declaration statement.
- Sometimes defines a typedef but doesn't use it. This applies to `26396.c`, `12281.c`, `9095.c`.

1.7 [60%] Init-Decl transformations 0 **BAD**

Description

Moves a declaration for a control statement if defined outside into the control statement.

Executable: `init_decl_transformer`

Options:

- strategy: `move_in_decls`

Example:

```
// test/21427.c
// Before transformation
int i;
for (i = 0; i < 7; i++) { \* ... *\ }

// After transformation
for (int i = 0; i < 7; i++) { \* ... *\ }
```

Total number of transformations in test/validation set: 229/210

Evaluation

Transformation	Good	Bad
`Init-Decl_transformations_0`	21427.c	23769.c
	5908.c	14741.c
	17357.c	6829.c
	16227.c	14024.c
	14654.c	20320.c
	3293.c	4341.c
	19999.c	4166.c
	20019.c	23872.c
	17513.c	
	25413.c	
	4766.c	
	18202.c	

- Introduces usage of undeclared variable errors due to existing code outside the loop scope that depends on the control variable. This applies to `23769.c`, `14741.c`, `6829.c`, `14024.c`, `4341.c`, `4166.c`, `23872.c`.

```
// Before transformation
usigned i;

for (i = 0; i < 10; i++)
  foo();

for (i = 0; i < 10; i++)
  bar();

// After transformation
for (unsigned i = 0; i < 10; i++)
  foo();

for (i = 0; i < 10; i++)        /* Should produce errors due to undeclared variable i */
  bar();
```

- Produced illegal syntax. This applies to `20320.c`.

```
// test/20320.c
// Before transformation
int sY = 0;
```

```
for (sY = 0; sY < height; dY++, sY++) { /* ... */ }

// After transformation
for (int sY = 0 = 0; sY < height; dY++, sY++) { /* ... */ }
```

1.8 [100%] Init-Decl transformations 1 GOOD

Description

Moves a declaration for a control variable if defined within control statement outside of the control statement.

Executable: `init_decl_transformer`

Options:

- strategy: `move_out_decls`

Example:

```
// test/7187.c
// Before transformation
for (int i = 0; i < frames_needed; i++) { \* ... *\ }

// After transformation
int i;
for (i = 0; i < frames_needed; i++) { \* ... *\ }
```

Total number of transformations in test/validation set: 0/1

Evaluation

Transformation	Good	Bad
`Init-Decl_transformations_0`	7187.c*	

Table 3: The star (*) symbol indicates that the transformation is from the validition set.

1.9 [100%] Input interface transformations 2 GOOD

Description

Instead of reading the input from stdin, the input is retrieved from a file.

Executable: `input_cstyle_transformer`

Options:

- strategy: freopentostdin

Example:

```
freopen("replace-me.in", "r", stdin);
```

The transformation inserts the code snippet above to the start of function body.

Total number of transformations in test/validation set: 17/9

Evaluation

Transformation	Good	Bad
`Input_interface_transformations_2`	15589.c	
	14679.c	
	1487.c	
	10190.c	
	4221.c	
	26158.c	
	3274.c	
	24222.c	
	19966.c	
	11113.c	
	24294.c	
	13051.c	
	8209.c	
	20197.c	
	9922.c	
	11516.c	
	17350.c	
	5874.c*	
	7353.c*	
	8065.c*	
	11628.c*	

Table 4: The star (*) symbol indicates that the transformation is from the validition set.

1.10 [15%] Literal transformations **BAD**

Description

Substitutes a return statement returning an integer literal, by a statement that returns a variable. The new variable is declared by the transformer and initialized accordingly.

Executable: `return_transformer`

Options:

- strategy: `S_IntLiteral`

Example:

```
// Before transformation
return 0;

// After transformation
int foo = 0;
return foo;
```

Total number of transformations in test/validation set: 668/678

Evaluation

Transformation	Good	Bad
`Literal_transformations`	22425.c	22844.c
	9510.c	265.c
	19724.c	24881.c
		17913.c
		24230.c
		12596.c
		15252.c
		14199.c
		9210.c
		4017.c
		21219.c
		3117.c
		26367.c
		5429.c
		11488.c
		5093.c
		8254.c

- No `-all` flag. Only applies transformation to a single return statement.
- Excessive removal of code. This applies to `22844.c`, `265.c`, `24881.c`, `17913.c`, `24230.c`, `12596.c`, `15252.c`, `14199.c`, `9210.c`, `4017.c`, `21219.c`, `3117.c`, `26367.c`, `5429.c`, `11488.c`, `5093.c`, `8254.c`.

1.11 [100%] Output interface transformations 3 GOOD

Description

Instead of printing the output to a file, the output is written directly to stdout (e.g. cout or printf).

Executable: `output_cstyle_transformer`

Options:

- strategy: `stdouttofreopen`

Example:

```
// Transformation adds following to the beginning of the function
freopen("replace-me.out", "w", stdout);
```

Total number of transformations in test/validation set: 17/9

Evaluation

Transformation	Good	Bad
`Output_interface_transformations_3`	3274.c	
	26158.c	
	20197.c	
	10190.c	
	15589.c	
	17350.c	
	24294.c	
	11113.c	
	8209.c	
	1487.c	
	19966.c	
	11516.c	
	24222.c	
	14679.c	
	4221.c	
	9922.c	
	13051.c	
	11628.c*	
	20992.c*	
	18342.c*	
	8065.c*	

Table 5: The star (*) symbol indicates that the transformation is from the validition set.

1.12 [0%] Unused code transformations 1 **BAD**

Description

Removes functions or global variables that are not used in the code.

Executable: `unused_declaration_transformer`

Options:

- strategy: `remove_local_decl`

Total number of transformations in test/validation set: 2123/2149

Evaluation

Transformation	Good	Bad
`Unused_code_transformations_1`		26173.c
		12362.c
		8878.c
		15373.c
		25868.c
		9150.c
		27169.c
		19434.c
		3731.c
		9943.c
		15492.c
		23912.c
		7670.c
		25413.c
		18015.c
		11631.c
		16752.c
		15517.c
		3227.c
		8112.c

- Removes function parameters and the closing parenthesis, resulting in invalid syntax.

```c
// test/26173.c
// Before transformation
static int hwupload_query_formats(AVFilterContext *avctx);

// After transformation
static int hwupload_query_formats(
```

- Removes used identifier declarations.

```c
// Before transformation
int err, i;
// ...
i += 1;

// after transformation
int err;
// ...
i += 1; // This will cause an error
```

- Or, removes random statements or large chunks of code.

1.13 [0%] Unused code transformations 2 **BAD**

Description

Removes functions or global variables that are not used in the code.

Executable: `unused_declaration_transformer`

Option:

- strategy: `remove_global_decl`

Total number of transformations in test/validation set: 2123/31

Evaluation

Transformation	Good	Bad
`Unused_code_transformations_2`		7762.c
		22421.c
		17030.c
		164.c
		20617.c
		23140.c
		12730.c
		8946.c
		8543.c
		5789.c
		20457.c
		3330.c
		21030.c
		27000.c
		8197.c
		12686.c
		9174.c
		17548.c
		6844.c
		372.c

- Removes function return types, resulting in invalid syntax.

```
// test/7762.c
// Before transformation
static int coroutine_fn is_allocated_base(BlockDriverState *top, /* ... */);

// After transformation
is_allocated_base(BlockDriverState *top, /* ... */);
```

- Removes left side of assignment.

```

// test/22421.c
// Before transformation
uint8_t b1 = cpu_ldub_code(env, dc->pc + 1);
// After transformation
= cpu_ldub_code(env, dc->pc + 1);
```

- Removes function modifiers.

```
// test/20617.c
// Before transformation
static av_always_inline av_flatten void h264_loop_filter_chroma_c(uint8_t *pix, int xstride,
    int ystride, int alpha, int beta, int8_t *tc0);
// After transformation
```

```
void h264_loop_filter_chroma_c(uint8_t *pix, int xstride, int ystride, int alpha, int beta,
    int8_t *tc0);
```

1.14 [95%] While statement transformations **BAD**

Description

Replaces a while-statement by an equivalent for-statement.

Executable: `while_transformer`

Example

```
// test/4681.c
// Before transformation
while (buf_size > 0) { /* ... */ }

// After transformation
for(; buf_size > 0;){ /* ... */ }
```

Total number of transformations in test/validation set: 82/86

Evaluation

Transformation	Good	Bad
`While_statement_transformations`	4681.c	14588.c
	3348.c	
	17904.c	
	20903.c	
	17308.c	
	11553.c	
	14306.c	
	16795.c	
	21822.c	
	17723.c	
	17412.c	
	18095.c	
	20918.c	
	20019.c	
	16808.c	
	9864.c	
	20301.c	
	8046.c	
	25925.c	

- No `-all` flag. Only applies the transformation to a single while statement.
- Moved loop control variable initialization into the control statement when code outside loop scope depends on an it being initialized. This applies to `14588.c`.

```
// test/14588.c
// Before transformation
int len = 0, height = avctx->height, width = avctx->width, x, y;
// ... code that depends on initialized len ...
while (len > 0) { /* ... */ }

// After transformation
int len, height = avctx->height, width = avctx->width, x, y;
// ... code that depends on initialized len ...
for(len = 0; len > 0;){ /* ... */ }
```

2 NatGen's code transformers **TRANSFORMER**

Contents

GitHub: https://github.com/saikat107/NatGen

NatGen's code transformers do not preserve formatting, including indentation and comments. Additionally, it surrounds text within single or double quotes with spaces, which may result in illegal syntax or change the semantics of the code. This issue has been addressed by removing all transformations that include quotes. Across all generated transformations by NatGen, this affects about half of the them.

2.1 [59%] Confusion remover **BAD**

Description

Removes confusing code including replacing ternary operators with if-else statements.

Example:

```
// Before transformation
return s->n_frames > 0 ? *out_size : size;


// After transformation
if (s->n_frames > 0) {
  return *out_size;
} else {
  return size;
}
```

Total number of transformations in test/validation set: 38/43

Evaluation

Transformation	Good	Bad
ConfusionRemover	1290.c	15486.c
	15615.c	10364.c
	3228.c	7558.c
	18474.c	11080.c
	24375.c	2280.c
	26100.c	10765.c
	14144.c	13969.c
	17308.c	13311.c
	16767.c	22065.c
	13316.c	
	1487.c	
	10180.c	
	14673.c	

- Removes member access operator ('.'). This affects `15486.c`, `10364.c`, `7558.c`, `11080.c`, `2280.c`, and `13311.c`.

```
// Before transformation
if (iscsilun->bl.max_xfer_len) { \* ... *\ }

// After transformation
```

```
if (iscsilun->blmax_xfer_len) { \* ... *\ }
```

- Attempts to assign to a constant variable outside of its declaration. This applies to `13969.c`.

```
// test/10765.c
// Before transformation
const int dim = BT_PAIR ? 2 : 4;

// After transformation
const int dim;
if (BT_PAIR) {
  dim = 2;
 } else {
  dim = 4;
 }
```

- Removes closing brackets. This applies to `13969.c`, and `22065.c`.

```
// test/13969.c
// Before transformation
  {
    // ...
    return prev ? NULL : internal->h;
  }

// After transformation
{
  // ...
  if (prev) {
    return NULL;
  } else {
    return internal->h;
  }
  // <- Missing closing bracket
```

2.2 [55%] Dead code inserter **BAD**

Description

Inserts random code that semantically does not change the program.

```
// Before transformation
BlockDriverState *bs = opaque;
qemu_co_enter_next(&bs->throttled_reqs[1]);

// After transformation
BlockDriverState *bs = opaque;
if (false) {                                  /* Dead code */
  qemu_co_enter_next(&bs->throttled_reqs[1]); /* Dead code */
}                                             /* Dead code */
qemu_co_enter_next(&bs->throttled_reqs[1]);
```

Listing 1: Example of a possible transformation inspired by a real transformation.

Total number of transformations in test/validation set: 1462/1432

Evaluation

Transformation	Good	Bad
DeadCodeInserter	13897.c	17258.c
	26251.c	3933.c
	5219.c	13149.c
	9174.c	22955.c
	3658.c	19274.c
	305.c	10161.c
	677.c	5607.c
	19998.c	5277.c
	25872.c	12429.c
	9607.c	
	10849.c	

- Changes if-statement's body. This applies to `17258.c`, `13149.c`.

```
// test/17258.c
// Before transformation
if (!l->has_value) {
  // if statement body
  // ...
 }

// After the transofmation
if (!l->has_value)
  if (_i_0 < _i_0) {
    {
      tcg_out_reloc(s, s->code_ptr, R_AARCH64_JUMP26, label_index, 0);
      tcg_out_goto_noaddr(s);
    }
  }
{ // <- Previous body of the if statement is now outside the if statement
  // if statement body
  // ...
}
```

- Usage of undeclared variable. This applies to `3933.c`, `22955.c`, `19274.c`, `10161.c`, `5607.c`, `12429.c`.

```
// test/3933.c
// Transformation adds:
while (_i_4 > _i_4) { // Transformation has not declared the used _i_4 variable
```

```
 {
   tcg_out_dat_reg(s, cond, opc, dst, lhs, rhs, SHIFT_IMM_LSL(0));
 }
}
```

- Introduces case statement without a switch statement. This applies to `5277.c`.

```
// test/5277.c
// Transformation adds:
while (false) {
 case MATROSKA_TRACK_ENCODING_COMP_LZO:
   do {
     olen = pkt_size *= 3;
     pkt_data = av_realloc(pkt_data, pkt_size + AV_LZO_OUTPUT_PADDING);
     result = av_lzo1x_decode(pkt_data, &olen, data, &isize);
   } while (result == AV_LZO_OUTPUT_FULL && pkt_size < 10000000);
   if (result)
     goto failed;
   pkt_size -= olen;
   break;
 }
```

2.3 [60%] For while transformer **BAD**

Description

Converts for loops into while loops or vice versa.

Example:

```
// test/6543.c
// Before transformation
for (i = 0; i < nb_desc; i++) {

// After transformation
i = 0;
while (i < nb_desc) {
  // ...
  i++;
 }
```

Total number of transformations in test/validation set: 439/441

Evaluation

Transformation	Good	Bad
ForWhileTransformer	26435.c	13130.c
	13859.c	6543.c
	6999.c	24606.c
	18426.c	7782.c
	21938.c	1476.c
	17678.c	23872.c
	674.c	22413.c
	16842.c	18194.c
	23769.c	
	11423.c	
	9607.c	
	265.c	

- Removes the member access operator ('.'). This applies to `13130.c`, `24606.c`, `18194.c`, `1476.c`, `23872.c`, and `22413.c`.

```
// test/13130.c
// Before transformation
u.l = qemu_get_be64(f);
env->fpr[i] = u.d;

// After transformation
ul = qemu_get_be64(f);
env->fpr[i] = ud;
```

- Replaces if statement's body, similarly to the previous transformation. This applies to `6543.c` and `7782.c`.

2.4 [85%] Operand swap **BAD**

Description

Swaps operands.

```
// Before transformation
if (foo == bar) { ... }

// After transformation
if (bar == foo) { ... }
```

Listing 2: Handwritten example of a possible transformation.

Total number of transformations in test/validation set: 857/847

Evaluation

Transformation	Good	Bad
OperandSwap	21261.c	22082.c
	24331.c	16771.c
	9846.c	25223.c
	4539.c	
	15521.c	
	17048.c	
	10180.c	
	26552.c	
	25404.c	
	1533.c	
	2187.c	
	11985.c	
	21622.c	
	7393.c	
	26435.c	
	13468.c	
	16385.c	

- Replaces NULL with an identifier. Applies to `22082.c` and `16771.c`, and `25223.c`.

```
// test/22082.c
// Before transformation
assert(bs != NULL);

// After transformation
assert(bs != bs);
```

2.5 [76%] Var renamer **BAD**

Description

Renames variables (and functions).

Example:

```
// Before transformation
int start_y, start_x, end_y, end_x, src_y_add = 0;

// After transformation
int start_y, VAR_0, end_y, end_x, VAR_2 = 0;
```

Total number of transformations in test/validation set: 1501/1472

Evaluation

Transformation	Good	Bad
VarRenamer	25566.c	5201.c
	9707.c	2300.c
	11080.c	7885.c
	20105.c	24488.c
	9574.c	14612.c
	18624.c	
	11141.c	
	10374.c	
	21092.c	
	18477.c	
	11423.c	
	20410.c	
	6519.c	
	24162.c	
	26552.c	
	2832.c	

- Renames calls to functions and usage of global variables.
- Does not update all instances of the renamed identifier. This applies to `5201.c`.
- Renames type instead of identifier. This applies to `2300.c`, `7885.c`.

```
// test/2300.c
// Before transformation
static av_always_inline void emulated_edge_mc(uint8_t *buf, const uint8_t *src, /* ... */);

// After transformation
static av_always_inline VAR_1 emulated_edge_mc(uint8_t *buf, const uint8_t *src, /* ... */);
```

- Inserts spaces between the pointer operator's characters (" - > "). This applies to `24488.c`.

```
// test/24488.c
// Before transformation
#if CONFIG_SMALL h264_filter_mb_fast_internal(h, sl, mb_x, mb_y, img_y,
img_cb, img_cr, linesize,
  uvlinesize, h->pixel_shift);
// After transformation
#if CONFIG_SMALL h264_filter_mb_fast_internal(h, sl, mb_x, mb_y, img_y, VAR_2,
img_cr, linesize, uvlinesize,
  h - > VAR_1);
```

- Renames an identifier and a variable's member of the same name. This applies to `14612.c`.

```
// test/14612.c
// Before transformation
AVIOContext *pb = s->pb;

// After transformation
AVIOContext *VAR_5 = s->VAR_5;
```

3 LimitsOfML4Vuln's code transformers **TRANSFORMER**

Contents

GitHub: https://github.com/niklasrisse/LimitsOfML4Vuln

LimitsOfML4Vuln's code transformations include two slightly different versions of the same code transformers for different datasets. The considered transformations have been generated using the transformations for CodeXGLUE.

3.1 [80%] Function Parameter Renaming (t1) **BAD**

Description

Renames function parameters, and refactors all usages of the parameters.

Example:

```
// Before transformation
void write(int *addr, int value){
  *addr = value;
}

// After transformation
void write(int *ax, int ot){
  *ax = ot;
}
```

Total number of transformations in test/validation set: 2728/2730

Evaluation

Transformation	Good	Bad
t1	9630.c	17513.c
	599.c	7828.c
	22365.c	19987.c
	2133.c	4091.c
	22906.c	
	26481.c	
	19431.c	
	26069.c	
	5314.c	
	5624.c	
	15534.c	
	16628.c	
	19274.c	
	7150.c	
	23277.c	
	19976.c	

- Reused the same identifier for a parameter. This applies to `17513.c`.

```c
// test/17513.c
// After transformation
static void quantize_and_encode_band_cost_ZERO_mips(struct AACEncContext *va, /* ... */,
                                                     const float *va, int jt, int xf, /* ...
                                                         */)
{ /* ... */ }
```

- Renames void in the function signature, which indicates no parameters. This applies to `7828.c`.

```c
// test/7828.c
// Before transformation
GSource *iohandler_get_g_source(void)
{ /* ... */ }

// After transformation
GSource *iohandler_get_g_source(cb)
{ /* ... */ }
```

- Refactors member variable of same name as the renamed parameter. This applies to `19987.c`.

```c
// test/19987.c
// Before transformation
static void cpu_devinit(const char *cpu_model, unsigned int id,
                        uint64_t prom_addr, qemu_irq **cpu_irqs)
{
  /* ... */
  env->prom_addr = prom_addr;
  /* ... */
}
// After transformation
static void cpu_devinit(const char *vi, unsigned int qv,
                        uint64_t uv, qemu_irq **of)
{
  /* ... */
  env->uv = uv;
  /* ... */
}
```

- Fails to update usage of all renamed identifiers. This applies to `4091.c`.

3.2 [0%] Function Parameter Reordering (t2) **BAD**

Description

Reorders function parameters.

Total number of transformations in test/validation set: 1790/1812

Evaluation

Transformation	Good	Bad
t2		3748.c
		22956.c
		15301.c
		19554.c
		26496.c
		20148.c
		17700.c
		12940.c
		12357.c
		26154.c
		21100.c
		10670.c
		2594.c
		13859.c
		25958.c
		25370.c
		9548.c
		10520.c
		14745.c
		6927.c

- Cuts off a large chunk of code. This applies to all examined transformations.

```
// test/3748.c
// Before transformation
int ff_h264_check_intra_pred_mode(H264Context *h, int mode, int is_chroma)
{ /* ... */ }

// After transformation
int ff_h264_check_intra_pred_mode(int is_chroma, int mode, H264Context *h)
  if (mode > 6U
      // It cuts off here ...
```

3.3 [85%] Function Renaming (t3) **BAD**

Description

Renames the function.

Example:

```
// Before transformation
static inline uint32_t insn_get( ... ) { ... }

// After transformation
static inline uint32_t mg( ... ) { ... }
```

Listing 3: Shortended real-world example of this type of transformation.

Total number of transformations in test/validation set: 2576/2578

Evaluation

Transformation	Good	Bad
t3	17009.c	21030.c
	17597.c	17030.c
	12789.c	11988.c
	6251.c	
	26435.c	
	24668.c	
	13768.c	
	12949.c	
	14741.c	
	4041.c	
	13513.c	
	6408.c	
	1909.c	
	3070.c	
	7762.c	
	2188.c	
	1894.c	

- Any text containing the function name as a part of it is also renamed. This has been encountered in `21030.c`, `17030.c`, `11988.c`.

```
// test/21030.c
// Before transformation
int attribute_align_arg avcodec_encode_audio(AVCodecContext *avctx, /* ... */);
av_log(avctx, AV_LOG_ERROR, "avcodec_encode_audio() does not " /* ... */);
/* ...
 * this is needed because the avcodec_encode_audio() API does not have
 ... */
ret = avcodec_encode_audio2(avctx, &pkt, frame, &got_packet);

// After transformation
int attribute_align_arg gx(AVCodecContext *avctx, /* ... */);
av_log(avctx, AV_LOG_ERROR, "gx() does not " /* ... */);
/* ...
 * this is needed because the gx() API does not have
 ... */
ret = gx2(avctx, &pkt, frame, &got_packet);
```

3.4 [100%] Insert Unexecuted Code (t4) **GOOD**

Description

Inserts code that is never executed.

Adds the following function to the end of the input file:

```
void helpfunc() {
  while (false) {
    break;
    break;
    // ... (a lot of breaks)
    break;
  }
}
```

Total number of transformations in test/validation set: 2732/2732

Evaluation

Transformation	Good	Bad
t_4	7119.c	
	10868.c	
	2246.c	
	18781.c	
	25081.c	
	6727.c	
	20738.c	
	4315.c	
	1508.c	
	4209.c	
	26087.c	
	3184.c	
	17596.c	
	22601.c	
	21383.c	
	15173.c	
	20507.c	
	14090.c	
	10295.c	
	12243.c	

- Transformation introduces an additional function.

3.5 [100%] Insert Comment (t5) GOOD

Description

Inserts a comment.

Example:

At the end of the file inserts a multi-line comment, spanning a single line, with the following content:

```
/*break; break; break; ... break;*/
```

Total number of transformations in test/validation set: 2732/2732

Evaluation

Transformation	Good	Bad
t_5	17735.c	
	3182.c	
	10322.c	
	11046.c	
	1119.c	
	20623.c	
	9896.c	
	18412.c	
	22949.c	
	18279.c	
	18596.c	
	4681.c	
	1508.c	
	12130.c	
	1883.c	
	25095.c	
	27254.c	
	6241.c	
	8357.c	
	4411.c	

3.6 [70%] Move Function Body into a Separate Function (t6) BAD

Description

Moves the body of a function into a separate function, called `helper_func`, and calls it from the original function.

```
// Before transformation
int add(int x, int y) {
  return x + y;
}

// After transformation
int helper_func(int x, int y) {
  return x + y;
}

int add(int x, int y) {
  return helper_func(x, y);
}
```

Total number of transformations in test/validation set: 2542/2537

Evaluation

Transformation	Good	Bad
t_6	1945.c	20761.c
	23425.c	13230.c
	27000.c	1508.c
	12036.c	23816.c
	9707.c	26873.c
	4178.c	14090.c
	9095.c	
	23626.c	
	7866.c	
	8466.c	
	22201.c	
	3610.c	
	21219.c	
	22601.c	

- Introduces a new function.
- Incorrect function call syntax when the function takes no arguments indicated by void in the parameter list. Encountered in `20761.c`.

```
// test/20761.c
// Before transformation
CharDriverState *qemu_chr_alloc(void)
{ /* ... */ }

// After transformation
CharDriverState helper_func(void)
{ /* ... */ }

CharDriverState *qemu_chr_alloc(void)
{
  return helper_func(void);
}
```

- Does not handle when function names are determined by macro. Encountered in `13230.c`, `1508.c`.

```
// test/13230.c
void FUNC(ff_simple_idct)(DCTELEM *block){
  // ...
  FUNC(idctRowCondDC)(block + i*8);
  // ...
  FUNC(idctSparseCol)(block + i);
  // ...
}

// After transformation
void helper_func(ff_simple_idct)(DCTELEM *block){
  // ...
  helper_func(idctRowCondDC)(block + i*8);
  // ...
  helper_func(idctSparseCol)(block + i);
}

void FUNC(ff_simple_idct)(DCTELEM *block)
{
  return helper_func(ff_simple_idct);
}
```

- Changes all text containing the function name. Encountered in `23816.c`, `14090.c`.

```
// test/23816.c
// Before transformation
static void conditional_branch(DBDMA_channel *ch)
{
  // ...
  DBDMA_DPRINTF("conditional_branch\n");
  // ...
}


// After transformation
static void helper_func(DBDMA_channel *ch)
{
  // ...
  DBDMA_DPRINTF("helper_func\n");
  // ...
}

static void conditional_branch(DBDMA_channel *ch)
{
  return helper_func(ch);
}
```

- Incorrect return type. Encountered in `26873.c`.

```

// test/26873.c
// Before transformation
static void *thread_func(void *p) { /* ... */ }

// After transformation
static void helper_func(void *p) { /* ... */ }

static void *thread_func(void *p)
{
  return helper_func(p); // <- Returns void instead of void *
}

```

3.7 [100%] Insert white space (t7) GOOD

Description

Inserts white spaces.

Adds a lot of space after the opening curly brace that defines the body of a function. The differences between the original and the transformed source codes are not visible to the naked eye.

Total number of transformations in test/validation set: 2732/2732

Evaluation

Transformation	Good	Bad
t_7	7125.c	
	17043.c	
	4025.c	
	18077.c	
	7814.c	
	21995.c	
	164.c	
	12846.c	
	26396.c	
	20189.c	
	12668.c	
	8534.c	
	20131.c	
	16256.c	
	244.c	
	18556.c	
	1040.c	
	13155.c	
	14665.c	
	26333.c	

3.8 [0%] Define Additional Void Function and Call it From the Function (t8) BAD

Description

Adds a void function and calls it from the original function.

From within the original function, calls `help_func();` and adds the following function to the end of the file:

```
void helpfunc() {
  return;
  return;
  return;
  // ... (a lot of returns)
  return;
}
```

Total number of transformations in test/validation set: 2732/2732

Evaluation

Transformation	Good	Bad
t_8		26315.c
		20305.c
		11624.c
		3205.c
		24754.c
		5250.c
		2148.c
		7472.c
		9036.c
		13001.c
		5171.c
		9072.c
		22426.c
		26836.c
		8835.c
		20086.c
		21576.c
		21583.c
		21061.c
		15537.c

- Introduces a new function.
- Calls `help_func` from within the main function, but the added function is called `helpfunc()`. Encountered in all transformations.

```
void original_func(){
  help_func();
}

void helpfunc(){
  return;
}
```

3.9 [100%] Remove all comments (t9) GOOD

Description

Removes all comments.

Total number of transformations in test/validation set: 1341/1340

Evaluation

Transformation	Good	Bad
t_9	9349.c	
	24292.c	
	4835.c	
	1318.c	
	11978.c	
	12243.c	
	21219.c	
	25859.c	
	26117.c	
	10502.c	
	16812.c	
	10351.c	
	18545.c	
	677.c	
	1883.c	
	3322.c	
	12642.c	
	19023.c	
	16205.c	
	1060.c	

3.10 [100%] Remove Newlines and Tabs (t12) (wo/ single line comments) GOOD

Description

Removes newlines and tabs.

Total number of transformations in test/validation set: 2383/2366

Evaluation

Transformation	Good	Bad
t_{12} (wo-single-line-comments)	20213.c	
	5099.c	
	11958.c	
	21296.c	
	402.c	
	23405.c	
	3375.c	
	25813.c	
	1713.c	
	4842.c	
	15082.c	
	10730.c	
	17439.c	
	3766.c	
	7109.c	
	4159.c	
	17513.c	
	10999.c	
	10504.c	
	1487.c	

4 RoPGen **TRANSFORMER**

Contents

GitHub: `https://github.com/RoPGen/RoPGen`

RoPGen works by transforming from a source style to a destination style. For each input file, we have considered only the most frequent source style and generated transformations for other destination styles.

4.1 [70%/80%/70%] Identifier Naming Method **BAD**

Description

Camel case, Pascal case, words separated by underscores, or identifiers starting with underscores.

- Style 0: Converts identifiers to camel case. E.g., `myVariableName`.
- Style 1: Converts identifiers to pascal case. E.g., `MyVariableName`.
- Style 2: Converts identifiers to words separated by underscores. E.g., `my_variable_name`.
- Style 3: Converts identifiers to identifiers starting with underscores. E.g., `_myVariableName`
- Style 4: Converts identifiers to start with a dollar sign. E.g. `$my_variable_name`.

Post-processing: Removed transformations including embedded assembly code snippets.

Total number of transformations in test/validation set: 1010/1005 + 1025/1023 + 33/30 + 1033/1028 + 37/29

Evaluation

Transformation	Good	Bad
Style 0 (wo-asm)	8487.c	7649.c
	5694.c	9727.c
	22276.c	13268.c
	23504.c	4213.c
	14199.c	20305.c
	5298.c	8949.c
	2297.c	
	16771.c	
	572.c	
	2996.c	
	25401.c	
	19728.c	
	21267.c	
	15723.c	
Style 1 (wo-asm)	18969.c	18512.c
	15420.c	21030.c
	1500.c	11268.c
	18583.c	21459.c
	21374.c	
	3290.c	
	11455.c	
	21092.c	
	8357.c	
	8457.c	
	19612.c	
	15115.c	
	10180.c	
	12434.c	
	10244.c	
	24424.c	
Style 2 (wo-asm)	25181.c	24294.c
	11427.c	3956.c
	15515.c	16190.c
	3.c	14694.c
	6243.c	19782.c
	16197.c	5582.c
	25081.c	
	11400.c	
	22065.c	
	14521.c	
	8988.c	
	18493.c	
	1404.c	
	8124.c	

- This transformation does not refactor usages of the renamed variable within embedded assembly code. Hence, we have excluded transformations that include embedded assembly code snippets. However, the exclusion was not exhaustive, and some transformations that include embedded assembly code have been included in the evaluation. We have not further attempted to remove these transformations as they also encounter other issues.
- The two final styles have not been evaluated as the encountered issues seem to be present in all the styles.

- Renames pointer members with the same identifiers as the refactored variables. This applies to

  Style 0: `7649.c`, `9727.c`, `13268`, `4213.c`, and `8949.c`.

  Style 1: `18512.c`, `21030.c`, and `11268.c`.

  Style 2: `3956.c`, `14694.c`, and `5582.c`.

```
// style 1
// test/7649.c
// Before transformation
AVFilterBufferRef *out_buf = outlink->out_buf;
outlink->out_buf = NULL;
// After transformation
AVFilterBufferRef *outBuf = outlink->outBuf;
outlink->outBuf = NULL;
```

- Does not refactor all usages of the renamed variable. This applies to

  Style 0: `20305.c`.

  Style 1: `21459.c`.

- Renames GCC extension keyword.

  Style 2: `24294.c`, and `19782.c`.

```
// test/24294.c
// Before transformation
int last_io __attribute__ ((unused)) = 0;
// After transformation
int last_io _attribute__ ((unused)) = 0;
```

- Does not refactor identifiers in embedded assembly code. This applies to

  Style 2: `16190.c`, `19782.c`.

4.2 [100%/95%] Access of Array/Pointer Elements GOOD:BAD

Description

Use the form of array indexes or pointers, e.g., `arr[i]` and `*(arr+i)`

- Style 0: Converts pointers to arrays. E.g., `*(chrSrcPtr+1)` to `chrSrcPtr[1]`. This example is grabbed from `13204.c`.
- Style 1: Does the opposite, converts arrays to pointers. E.g. `codebook_multiplicands[i]` to `*(codebook_multiplicands+`
  This example is grabbed from `10175.c`.

Total number of transformations in test/validation set: 3/6 + 152/148

Evaluation

Transformation	Good	Bad
Style 0	20022.c	
	8046.c	
	13204.c	
	17645.c*	
	6967.c*	
	1836.c*	
	19864.c*	
	25098.c*	
	25743.c*	
Style 1	18624.c	7558.c
	6829.c	
	27237.c	
	16727.c	
	22436.c	
	2115.c	
	8124.c	
	15001.c	
	8169.c	
	22956.c	
	19965.c	
	5826.c	
	4025.c	
	26435.c	
	10244.c	
	19595.c	
	2677.c	
	8273.c	
	3293.c	

Table 6: The star (*) symbol indicates that the transformation is from the validition set.

- Invalid syntax. This affects:

  Style 1: `7558.c`.

```
// style 1
// test/7558.c

// Before transformation
bands[i].scf_idx[ch][0] = get_bits(gb, 7) - 6;
// ...
bands[i].scf_idx[ch][0] = ((bands[i].scf_idx[ch][2] + t - 25) & 0x7F) - 6;
```

```
// After transformation
bands[i].*(*(scf_idx+ch)+0) = get_bits(gb, 7) - 6;
// ...
bands[i].*(*(scf_idx+ch)+0) = ((bands[i].*(*(scf_idx+ch)+2) + t - 25) & 0x7F) - 6;
```

4.3 [60%/67%] Location of Defining Local Variables **BAD**

Description

Local variables are defined at the begging of the variable scope, or each local variable is defined when used for the first time.

- Style 0: Defines variables at the begging of the begging of the variable scope.
- Style 1: Defines local variables when used for the first time.

Total number of transformations in test/validation set: 220/210 + 1838/1811

Evaluation

Transformation	Good	Bad
Style 0	10899.c	24018.c
	26437.c	22597.c
	3602.c	15002.c
	6935.c	5314.c
	5694.c	1804.c
	12278.c	7814.c
	20917.c	1136.c
	5582.c	7327.c
	16783.c	
	17953.c	
	22399.c	
	10414.c	
Style 1 (wo asm)	14228.c	18773.c
	599.c	17246.c
	23354.c	
	12255.c	

- Does not handle macros. Moves definition of variables to other macro scopes. This affects

  Style 0: `24018.c`, `22597.c`, `15002.c`, `5314.c`, `1804.c`, `7814.c`, `1136.c`, `7327.c`.

  Style 1: `18773.c`.

```
// Style 0
// test/24018.c
// Before transformation
int out_idx = 0;
#if 0 //decode order

// After transformation
#if 0 //decode order
int out_idx = 0;
```

- Does not handle labels. This affects

  Style 1: `17246.c`.

```
// My example
// Before transformation
void main(){
  int a = 0;    // 1. Declaration
  if (1)
    goto exit; // 2. GOTO label
  if (a == 1)
    a = 2;
 exit:
  return a;     // 3. Returns declared variable
}
```

```
// After transformation
int main(){
  if (1)
    goto exit; // 1. GOTO label

  int a = 1;
  if (a == 1)
    a = 2;

 exit:
  return a;   // 2. Variable a is not declared
}
```

4.4 [85%/100%] Location of Initializing Local Variables **BAD:GOOD**

Description

Local variables are initialized and declared in same statements, or in different statements.

- Style 0: Initializes variables in the same statement as they are defined.
- Style 1: Initializes variables in a separate statement.

Total number of transformations in test/validation set: 137/121 + 1639/1625

Evaluation

Transformation	Good	Bad
Style 0	10554.c	9922.c
	14612.c	
	5672.c	
	20741.c	
	17452.c	
	8900.c	
Style 1	1883.c	
	707.c	
	1817.c	
	10984.c	
	9039.c	
	122.c	
	11854.c	
	12667.c	
	8217.c	
	2142.c	
	17439.c	
	15589.c	
	21296.c	
	24400.c	
	13888.c	
	12686.c	
	19716.c	
	25265.c	
	27021.c	
	8900.c	

- **Style 0**
  - Transformation alters program behavior. This affects: `9922.c`.

```
// Example inspired by 9922.c
// Before transformation
int a;
while (whatever) {
  a = 123;
  // ...
 }
a = 10;
return a; // This will return 10


// After transformation
int a = 10;
while (whatever) {
  a = 123;
  // ...
 }
return a; // This will return 123
```

4.5 [100%/100%] Definition (and Initialization) of Multiple Variables with Same Types **GOOD**

Description

Multiple variables with same types are defined (and initialized) in a statement or in multiple statements.

- Style 0: Combines variable definitions of same type into a single statement.
- Style 1: Splits multiple variable definitions in a single statement into multiple statements.

Total number of transformations in test/validation set: 828/777 + 869/872

Evaluation

Transformation	Good	Bad
Style 0	20283.c	
	10539.c	
	8273.c	
	4029.c	
	11383.c	
	1500.c	
	8815.c	
	15094.c	
	7099.c	
	14658.c	
	15301.c	
	9025.c	
	4362.c	
	15515.c	
	794.c	
	599.c	
	9706.c	
	12099.c	
	9922.c	
Style 1	25413.c	
	14588.c	
	12281.c	
	8988.c	
	16602.c	
	18200.c	
	16197.c	
	16771.c	
	11795.c	
	657.c	
	5755.c	
	8900.c	
	3763.c	
	5864.c	
	26946.c	
	7099.c	
	10175.c	
	12894.c	
	26245.c	
	899.c	
	25472.c	

4.6 [100%/100%] Variable Assignment GOOD

Description

Multiple variable assignments are in a statement (e.g., `tmp=++i;`) or multiple statements (e.g., `++i; tmp=i`).

- Style 0: Combines multiple assignments in a single statement.
- Style 1: Splits multiple assignments into multiple statements.

Total number of transformations in test/validation set: 2/0 + 13/11

Evaluation

Transformation	Good	Bad
Style 0	18963.c	
	15094.c	
Style 1	25859.c	
	23193.c	
	2602.c	
	244.c	
	22426.c	
	21641.c	
	4896.c	
	4255.c	
	12789.c	
	11383.c	
	25937.c	
	24881.c	
	17120.c	
	6917.c*	
	9241.c*	
	19423.c*	
	16988.c*	
	11384.c*	
	7171.c*	
	7543.c*	

4.7 [95%] Memory Allocation **BAD**

Description

Static array allocation (e.g., `int arr[100];`) or dynamic memory allocation (e.g., `int *arr=malloc(100*sizeof(int));`).

Style 1: Transforms to dynamic memory allocation.

Side effect:

- Does not add the necessary include directive for `malloc`.
- Does not free the allocated memory.

Total number of transformations in test/validation set: 181/162

Evaluation

Transformation	Good	Bad
Style 1	17752.c	19892.c
	23747.c	
	16808.c	
	25505.c	
	24648.c	
	6125.c	
	26136.c	
	20005.c	
	17766.c	
	6524.c	
	24761.c	
	4801.c	
	2903.c	
	15368.c	
	26031.c	
	11361.c	
	7609.c	
	8955.c	
	2656.c	

- Produces invalid syntax. This affects: `19892.c`.

```
// test/19892.c
// Before transformation
unsigned char scratch[VQA_PREAMBLE_SIZE];

// After transformation
unsigned*char scratch = (unsigned*)malloc(sizeof(unsigned)*(VQA_PREAMBLE_SIZE));
```

4.8 [100%/90%] Loop Structures **BAD:GOOD**

Description

Use for structure or while structure.

- Style 0: Use `for` loops.
- Style 1: Use `while` loops.

Total number of transformations in test/validation set: 308/308 + 774/797

Evaluation

Transformation	Good	Bad
Style 0	4315.c	
	6535.c	
	4196.c	
	21219.c	
	18518.c	
	8534.c	
	9864.c	
	8846.c	
	13855.c	
	3001.c	
	9943.c	
	17314.c	
	9366.c	
	14911.c	
	16795.c	
	20903.c	
	6532.c	
	25925.c	
	566.c	
	6489.c	
Style 1	18457.c	18187.c
	4166.c	17412.c
	12616.c	
	121.c	
	18194.c	
	16693.c	
	7393.c	
	5908.c	
	7834.c	
	338.c	
	23050.c	
	9258.c	
	8085.c	
	22330.c	
	20237.c	
	15108.c	
	547.c	
	25530.c	

- **Style 1**: If the keyword `continue` is present in a loop, the conditional variable is not updated. This affects `18187.c`, and `17412.c`.

```
// Example
// Before transformation
```

```
for (i = 0; i < num_bs; i++) {
  if (i == 0) {
    // This will increment i
    continue;
  }
 }

// After transformation
i = 0;
while (i < num_bs)  {
  if (i == 0) {
    // This will not increment i
    continue;
  }
  i++;
 }
```

4.9 [95%/84%] Conditional Structures **BAD**

Description

Use conditional operator, if-else, or switch-case structure.

- Style 0: Converts ternary operators into if-else statements.

```
bool says_hi = false;

// Coalescing operator, a variation of the ternary operator
a = says_hi ? "Hello" : NULL;

// After applying style 0 transformation
if (says_hi){
  a = "Hello";
} else {
  a = NULL;
}
```

- Style 1: Converts switch statements into if-else.

Total number of transformations in test/validation set: 167/163 + 118/139

Evaluation

Transformation	Good	Bad
Style 0	23299.c	25245.c
	12095.c	
	25353.c	
	13462.c	
	9349.c	
	10672.c	
	9864.c	
	14741.c	
	24206.c	
	21938.c	
	3401.c	
	6008.c	
	26512.c	
	5673.c	
	4781.c	
	7670.c	
	7716.c	
	7808.c	
	26110.c	
Style 1	13988.c	3153.c
	27261.c	6556.c
	490.c	9509.c
	14521.c	
	7992.c	
	18462.c	
	7521.c	
	20840.c	
	6125.c	
	18931.c	
	9343.c	
	20518.c	
	9864.c	
	23245.c	
	25602.c	
	2895.c	

- **Style 0**
  - Missing spaces in variable declarations. This affects `25245.c`.

```
// Style 0
// test/25245.c
// Before transformation
const int16_t (*l1mv)[2] = l1ref0[i8] == 0 ? l1mv0 : l1mv1;
if (IS_SUB_8X8(sub_mb_type)) { /* ... */ }

// After transformation
if(l1ref0[i8] == 0){constint16_t(*l1mv)[2]=l1mv0;
 }else{constint16_t(*l1mv)[2]=l1mv1;
 }if (IS_SUB_8X8(sub_mb_type)) { /* ... */ }
```

- **Style 1**
  - Does not handle when macros or templates are used for inserting case statements. This affects `3153.c`.

```
// Style 1
// test/3153.c
// Before transformation
switch (c->srcFormat) {
 case PIX_FMT_Y400A:
   c->lumToYV12 = ff_yuyvToY_mmx;
   if (c->alpPixBuf)
     c->alpToYV12 = ff_uyvyToY_mmx;
   break;
 case PIX_FMT_YUYV422:
   c->lumToYV12 = ff_yuyvToY_mmx;
   c->chrToYV12 = ff_yuyvToUV_mmx;
   break;
 case PIX_FMT_UYVY422:
   c->lumToYV12 = ff_uyvyToY_mmx;
   c->chrToYV12 = ff_uyvyToUV_mmx;
   break;
 case PIX_FMT_NV12:
   c->chrToYV12 = ff_nv12ToUV_mmx;
   break;
 case PIX_FMT_NV21:
   c->chrToYV12 = ff_nv21ToUV_mmx;
   break;
   case_rgb(rgb24, RGB24, mmx);
   case_rgb(bgr24, BGR24, mmx);
   case_rgb(bgra,  BGRA,  mmx);
   case_rgb(rgba,  RGBA,  mmx);
   case_rgb(abgr,  ABGR,  mmx);
   case_rgb(argb,  ARGB,  mmx);
 default:
   break;
 }

// After transformation
if(c->srcFormat==PIX_FMT_Y400A)
  {c->lumToYV12 = ff_yuyvToY_mmx;
    if (c->alpPixBuf)
      c->alpToYV12 = ff_uyvyToY_mmx;
  }else if(c->srcFormat==PIX_FMT_YUYV422)
  {c->lumToYV12 = ff_yuyvToY_mmx;
    c->chrToYV12 = ff_yuyvToUV_mmx;
  }else if(c->srcFormat==PIX_FMT_UYVY422)
  {c->lumToYV12 = ff_uyvyToY_mmx;
    c->chrToYV12 = ff_uyvyToUV_mmx;
  }else if(c->srcFormat==PIX_FMT_NV12)
  {c->chrToYV12 = ff_nv12ToUV_mmx;
  }else if(c->srcFormat==PIX_FMT_NV21)
  {c->chrToYV12 = ff_nv21ToUV_mmx;
  }case_rgb(rgb24, RGB24, mmx);
case_rgb(bgr24, BGR24, mmx);
case_rgb(bgra,  BGRA,  mmx);
case_rgb(rgba,  RGBA,  mmx);
case_rgb(abgr,  ABGR,  mmx);
case_rgb(argb,  ARGB,  mmx);
 else
   { }
```

– Invalid syntax. Does not handle empty default which refers to the next case. This affects `6556.c`, `9509.c`.

```
// Style 1
// test/6556.c
// Before transformation
default:

case HTTPSTATE_SEND_DATA_TRAILER:
    /* last packet test ? */
    if (c->last_packet_sent)
        return -1;
    /* prepare header */
    av_write_trailer(&c->fmt_ctx);
    c->last_packet_sent = 1;
    break;
}

// After transformation
 else
   {av_write_trailer(&c->fmt_ctx);
     c->last_packet_sent = 1;

   }else if(c->state==HTTPSTATE_SEND_DATA_TRAILER)
      {/* last packet test ? */
        if (c->last_packet_sent)
          return -1;
        /* prepare header */
      }
```

4.10 [50%/100%] Compound `if` Statements **BAD:GOOD**

Description

Use a logical operator in an if condition (e.g., `if(a && b)`) or use multiple if conditions (e.g., `if(a){if(b){...}}`).

- Style 0: Transforms multiple nested if conditions into a single if condition using logical operators.
- Style 1: Transforms if conditions using logical operators into multiple nested if conditions.

Total number of transformations in test/validation set: 110/117 + 513/523

Evaluation

Transformation	Good	Bad
Style 0	9706.c	19724.c
	6385.c	21012.c
	9349.c	26777.c
	7967.c	12451.c
	21670.c	20433.c
	5190.c	23508.c
	17012.c	13172.c
	26836.c	9094.c
	13906.c	7766.c
Style 1	13504.c	
	7521.c	
	11742.c	
	1130.c	
	748.c	
	25310.c	
	17082.c	
	24626.c	
	25370.c	
	14588.c	
	11316.c	
	21925.c	
	8534.c	
	25574.c	
	26745.c	
	24759.c	
	17009.c	
	20647.c	
	20148.c	
	22167.c	

- The transformation changes the behavior of the code. This is the case for every transformation of style 0 that has an `else` statement.

Style 0: `19724.c`, `21012.c`, `26777.c`, `12451.c`, `20433.c`, `23508.c`, `13172.c, =9094.c`, and `7766.c`.

```
// Before transformation
if (a == 0){
  if (b == 0)
    return b;
} else if (c == 0) {
  return c;
}

// After transformation
if (a == 0 && b ==0){
  return b;
} else if (c == 0) {
  return c;
}
```

Listing 4: Before the transformation, if `a` is `0`, `c` will never be returned. After the transformation `c` may be returned even if `a` is `0`.