

Input: Function call code in function *func* at WyIL level

Output: A list of function call code, optimised by copy and deallocation analysers, includes pre-deallocation, the actual code of function call and post-deallocation

1: **Variables**

CopyAnalyser: Copy elimination analyser

DeallocAnalyser: De-allocation analyser

list: A list of optimised C code

aParam: Actual parameter at function call code

aParam_size: Size of actual parameter *aParam*

tmp_aParam: Temporary variable used to store the copy of *aParam*, e.g. *tmp_a_0* holds copied *a* at index of 0 in parameter list

CodeStore: A code store keeps track of the variable names of temporary parameters

macro: post-deallocation *macro* of *aParam* from deallocation analyser

2: **end Variables**

3: **procedure** OPTIMISEFUNCTIONCALL(*code*, *func*)

4: *list* = []

5: *ret* = function return variable at *code*

6: *callee* = the name of called function at *code*

7: *list.append*("PRE_DEALLOC(*ret*)")// Pre-Deallocation Macro on *ret*

8: **for each** *aParam* in *code* **do**// Beginning temporary variables

9: **if** (*aParam* is Array AND

10: ¬*CopyAnalyser.ISCOPYREMOVED*(*aParam*, *code*, *func*)) **then**

11: *tmp_aParam* ← *CodeStore*(*aParam*, *code*, *func*)

12: // Define temporary variable *tmp_aParam*

13: *list.append*(void* *tmp_aParam* = COPY(*aParam*);)

14: **end if**

15: **end for**// Ending declaration

16: // Actual code of function call

17: *list.append*("ret = *callee*("// Beginning function call statement

18: **for each** *aParam* in *code* **do**

19: **if** *CopyAnalyser.ISCOPYREMOVED*(*param*, *code*, *func*) **then**

20: // Pass *aParam* with false flag

21: *list.append*(*aParam*, *aParam_size*, false)

22: **else**

23: *macro* ← *DeallocAnalyser.CHOOSEPOSTMACRO*(*param*,
 code, *func*)// Get post deallocation *macro* of
 aParam

24: // Pass copied parameter *tmp_param*

25: **if** *macro* == CALLEE **then**

26: *list.append*(*tmp_aParam*, *aParam_size*, true)

27: **else**// CALLER

28: *list.append*(*tmp_aParam*, *aParam_size*, false)

29: **end if**

30: **end if**

31: **end for**

32: *list.append*(");")// Ending function call

33: // Post deallocation macro

34: **for each** *aParam* in *code* **do**// Beginning post deallocation macro

35: *macro* ← *DeallocAnalyser.CHOOSEPOSTMACRO*(*param*, *code*,
 func)

36: **switch** *macro* **do**

37: **case** RETAIN// Apply retain macro on *aParam*

38: *list.append*("RETAIN_DEALLOC_POST(*ret*, *aParam*)")

39: **case** RESET// Apply reset macro on *aParam*

40: *list.append*("RESET_DEALLOC_POST(*ret*, *aParam*)")

41: **case** CALLER// Apply caller macro on *tmp_aParam*

42: *list.append*("CALLER_DEALLOC_POST(*ret*, *tmp_aParam*)")

43: **case** CALLEE// Apply callee macro on *tmp_aParam*

44: *list.append*("CALLEE_DEALLOC_POST(*ret*, *tmp_aParam*)")

45: **end for**// Ending post macro

46: **return** *list*

47: **end procedure**