Control Flow in GCC Generated Compiler

Uday Khedker

(www.cse.iitb.ac.in/~uday)

GCC Resource Center,
Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

13 June 2014

Walking the Maze of a Large Code Base

- If you use conventional editors such as `vi` or `emacs`
  - Use cscope

    ```
    cd $SOURCE
    cscope -R
    ```

  - Use ctags

    ```
    cd $SOURCE
    ctags -R
    ```

    Make sure you use `exeburant-ctags`
- Or use IDE such as `eclipse`

gcc Driver Control Flow

```
main   /* In file gcc.c */
   validate_all_switches
   lookup_compiler
   do_spec
      do_spec_2
         do_spec_1  /* Get the name of the compiler */
      execute
         pex_init
         pex_run
            pex_run_in_environment
               obj->funcs->exec_child
```


gcc Driver Control Flow

```
main   /* In file gcc.c */
   validate_all_switches
   lookup_compiler
   do_spec
      do_spec_2
         do_spec_1  /*
      execute
         pex_init
         pex_run
            pex_run_in
               obj->fu
```

Observations

- All compilers are invoked by this driver
- Assembler is also invoked by this driver
- Linker is invoked in the end by default

cc1 Top Level Control Flow

```
main                    /* In file main.c */
|   toplev_main    /* In file toplev.c */
|   |   decode_options
|   |   do_compile
|   |   |   compile_file
|   |   |       lang_hooks.parse_file => c_common_parse_file
|   |   |       lang_hooks.decls.final_write_globals =>
|   |   |                                   c_write_global_declarations
|   |   |     targetm.asm_out.file_end
|   |   finalize
|   |
|
```


cc1 Top Level Control Flow

```
main                    /* In file main.c */
  toplev_main           /* In file toplev.c */
    decode_options
    do_compile
      compile_file
        lang_hooks                                  le
        lang_hooks
                                                    eclarations
        targetm.asm
    finalize
```

Observations

- The entire compilation is driven by functions specified in language hooks
- Not a good design!

cc1 Control Flow: Parsing for C

```
lang_hooks.parse_file => c_common_parse_file
   c_parse_file
      c_parser_translation_unit
         c_parser_external_declaration
            c_parser_declaration_or_fndef
               c_parser_declspecs /* parse declarations */
               c_parser_compound_statement
               finish_function    /* finish parsing */
                  c_genericize
                  cgraph_finalize_function
                  /* finalize AST of a function */
```


cc1 Control Flow: Parsing for C

```
lang_hooks.parse_file => c_common_parse_file
  c_parse_file
    c_parser_translation_unit
      c_parser_external_declaration
        c_parser_declaration_or_fndef
          c_parser_declspecs /* parse declarations */
          c_parser_compound_statement
          finish_function    /* finish parsing */
            c_genericize
            cgraph_finalize_function
            /* finalize AST of a function */
```

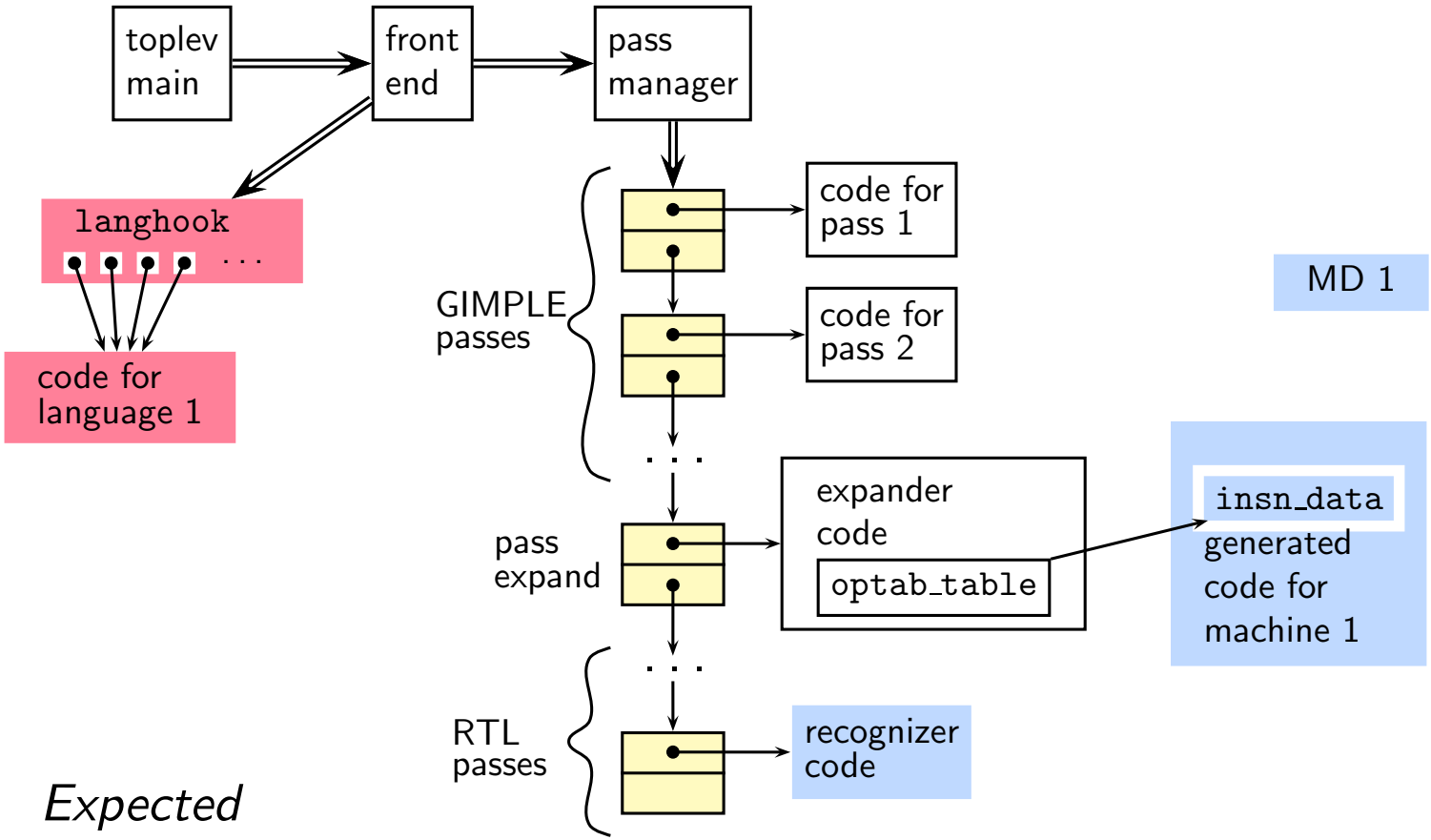

cc1 Control Flow: Parsing for C

```
lang_hooks.parse_file => c_common_parse_file
   c_parse_file
      c_parser_translation_unit
         c_parser_external_declaration
            c_parser_declaration_or_fndef
               c_pars                                    ns */
               c_pars
               finish                                    /
                  c_g
                  cgr
                  /*
```

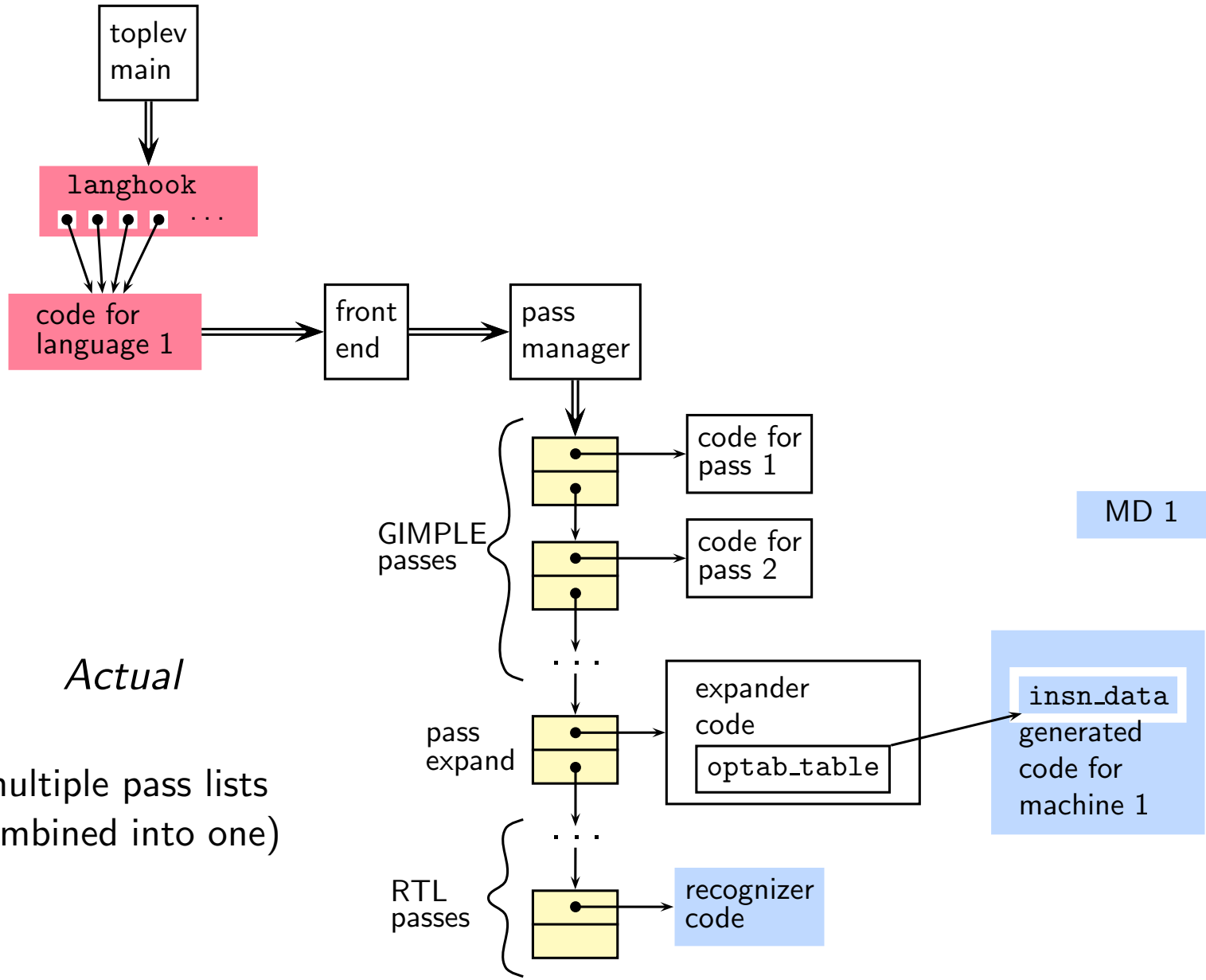
Observations

- GCC has moved to a recursive descent parser from version 4.1.0
- Earlier parser was generated using Bison specification

Expected Vs. Actual Schematic of the Front End



Expected Vs. Actual Schematic of the Front End



cc1 Control Flow: Lowering Passes for C

```
lang_hooks.decls.final_write_globals =>
                                        c_write_global_declarations
    cgraph_finalize_compilation_unit
       cgraph_analyze_functions      /* Create GIMPLE */
          cgraph_analyze_function
             gimplify_function_tree
                gimplify_body
                   gimplify_stmt
                      gimplify_expr
             cgraph_lower_function    /* Intraprocedural */
                tree_lowering_passes
                   execute_pass_list (all_lowering_passes)
```


cc1 Control Flow: Lowering Passes for C

```
lang_hooks.decls.final_write_globals =>
                                        c_write_global_declarations
    cgraph_finalize_compilation_unit
      cgraph_analyze_functions      /* Create GIMPLE */
        cgraph_analyze_function
          gimplify_function_tree
            gimplify_body
               gimplify_stmt
                  gimplify_expr
          cgraph_lower_function     /* Intraprocedural */
            tree_lowering_passes
               execute_pass_list (all_lowering_passes)
```


cc1 Control Flow: Lowering Passes for C

```
lang_hooks.decls.final_write_globals =>
                                    c_write_global_declarations
    cgraph_finalize_compilation_unit
       cgraph_analyze_functions      /* Create GIMPLE */
          cgraph_analyze_function
             gimplify
                gimpl
                   gi

             cgraph_l                                  ural */
                tree_
                   ex                                  asses)
```

Observations

- Lowering passes are language independent
- Yet they are being called from a function in language hooks
- Not a good design!

Organization of Passes

Order	Task	IR	Level	Pass data structure
1	Lowering	GIMPLE	Intra	`gimple_opt_pass`
2	Optimizations	GIMPLE	Inter	`simple_ipa_opt_pass`
3	Optimizations	GIMPLE	Inter	`ipa_opt_pass_d`
4	Optimizations	GIMPLE	Intra	`gimple_opt_pass`
5	RTL Generation	GIMPLE	Intra	`rtl_opt_pass`
6	Optimization	RTL	Intra	`rtl_opt_pass`

cc1 Control Flow: Optimization and Code Generation Passes

```
cgraph_analyze_function          /* Create GIMPLE */
...                              /* previous slide */
cgraph_optimize
   ipa_passes
      execute_ipa_pass_list(all_small_ipa_passes)  /*!in_lto_p*/
      execute_ipa_summary_passes(all_regular_ipa_passes)
      execute_ipa_summary_passes(all_lto_gen_passes)
      ipa_write_summaries
   execute_ipa_pass_list(all_late_ipa_passes)
   cgraph_expand_all_functions
       cgraph_expand_function
       /* Intraprocedural passes on GIMPLE,  */
       /* expansion pass, and passes on RTL. */
            tree_rest_of_compilation
               execute_pass_list (all_passes)
```

cc1 Control Flow: Optimization and Code Generation Passes

```
cgraph_analyze_function          /* Create GIMPLE */
...                              /* previous slide */
cgraph_optimize
  ipa_passes
    execute_ipa_pass_list(all_small_ipa_passes)  /*!in_lto_p*/
    execute_ipa_summary_passes(all_regular_ipa_passes)
    execute_ipa_summary_passes(all_lto_gen_passes)
    ipa_write_summaries
  execute_ipa_pass_list(all_late_ipa_passes)
  cgraph_expand_all_functions
    cgraph_expand_function
    /* Intraprocedural passes on GIMPLE,  */
    /* expansion pass, and passes on RTL. */
        tree_rest_of_compilation
           execute_pass_list (all_passes)
```


cc1 Control Flow: Optimization and Code Generation Passes

```
cgraph_analyze_function          /* Create GIMPLE */
...                              /* previous slide */
cgraph_optimize
   ipa_passes
      execute_ipa_pas                                  !in_lto_p*/
      execute_ipa_sum                                  ses)
      execute_ipa_sum
      ipa_write_summa
   execute_ipa_pass_l
   cgraph_expand_all_
       cgraph_expand_
       /* Intraproced
       /* expansion p
            tree_rest
               execut
```

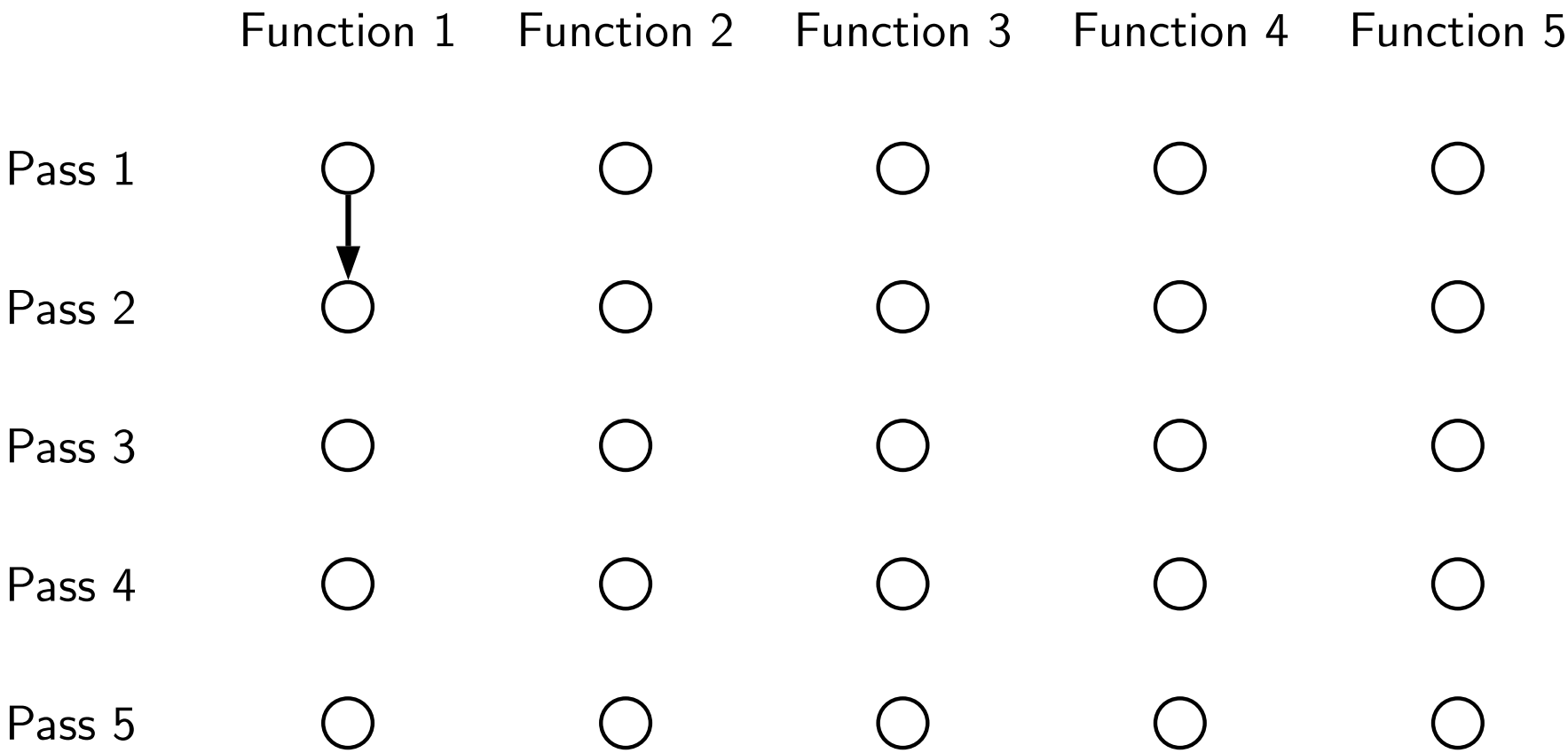
Observations

- Optimization and code generation passes are language independent
- Yet they are being called from a function in language hooks
- Not a good design!

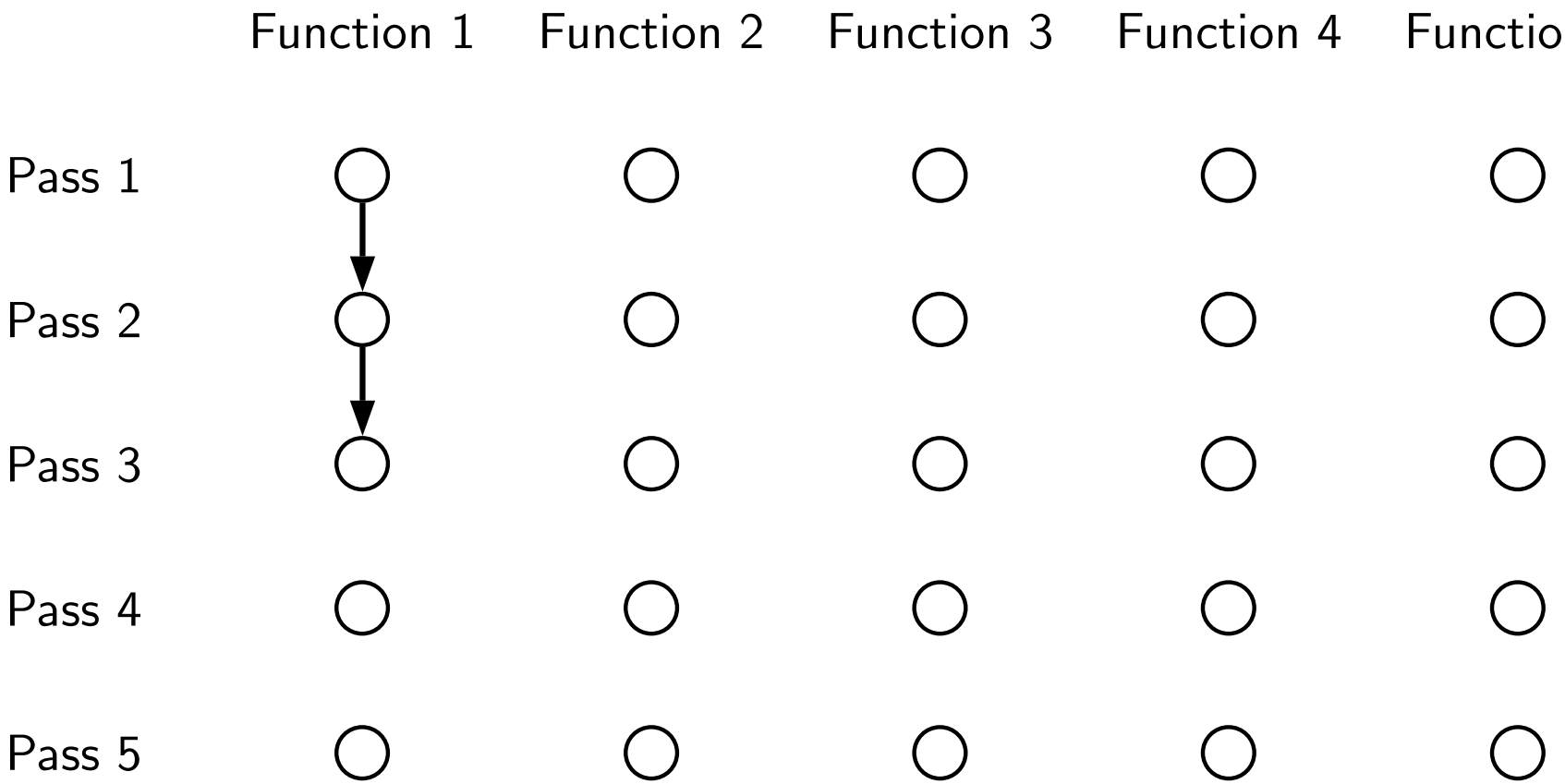
Execution Order in Intraprocedural Passes

	Function 1	Function 2	Function 3	Function 4	Function 5
Pass 1	○	○	○	○	○
Pass 2	○	○	○	○	○
Pass 3	○	○	○	○	○
Pass 4	○	○	○	○	○
Pass 5	○	○	○	○	○

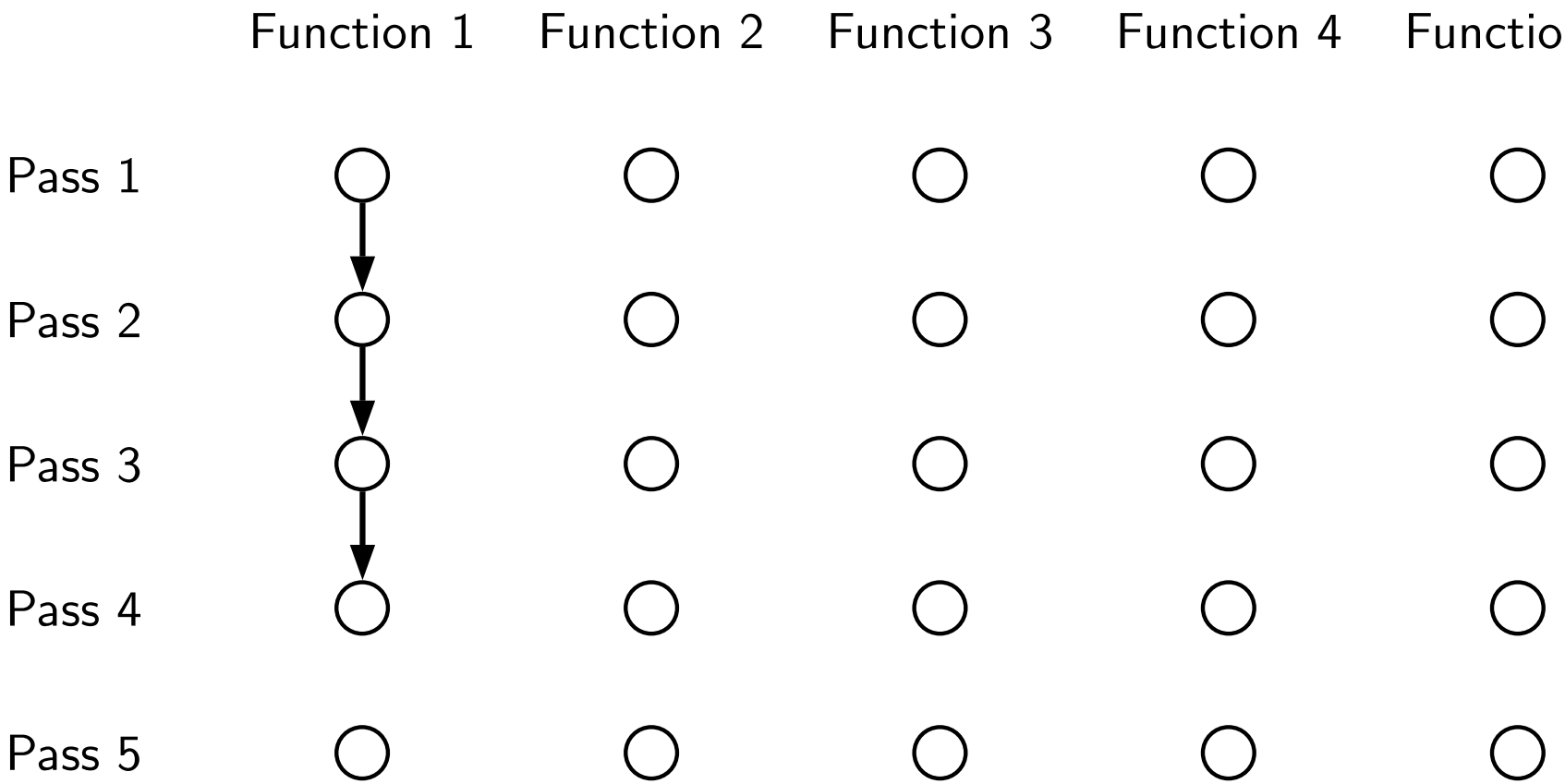
Execution Order in Intraprocedural Passes



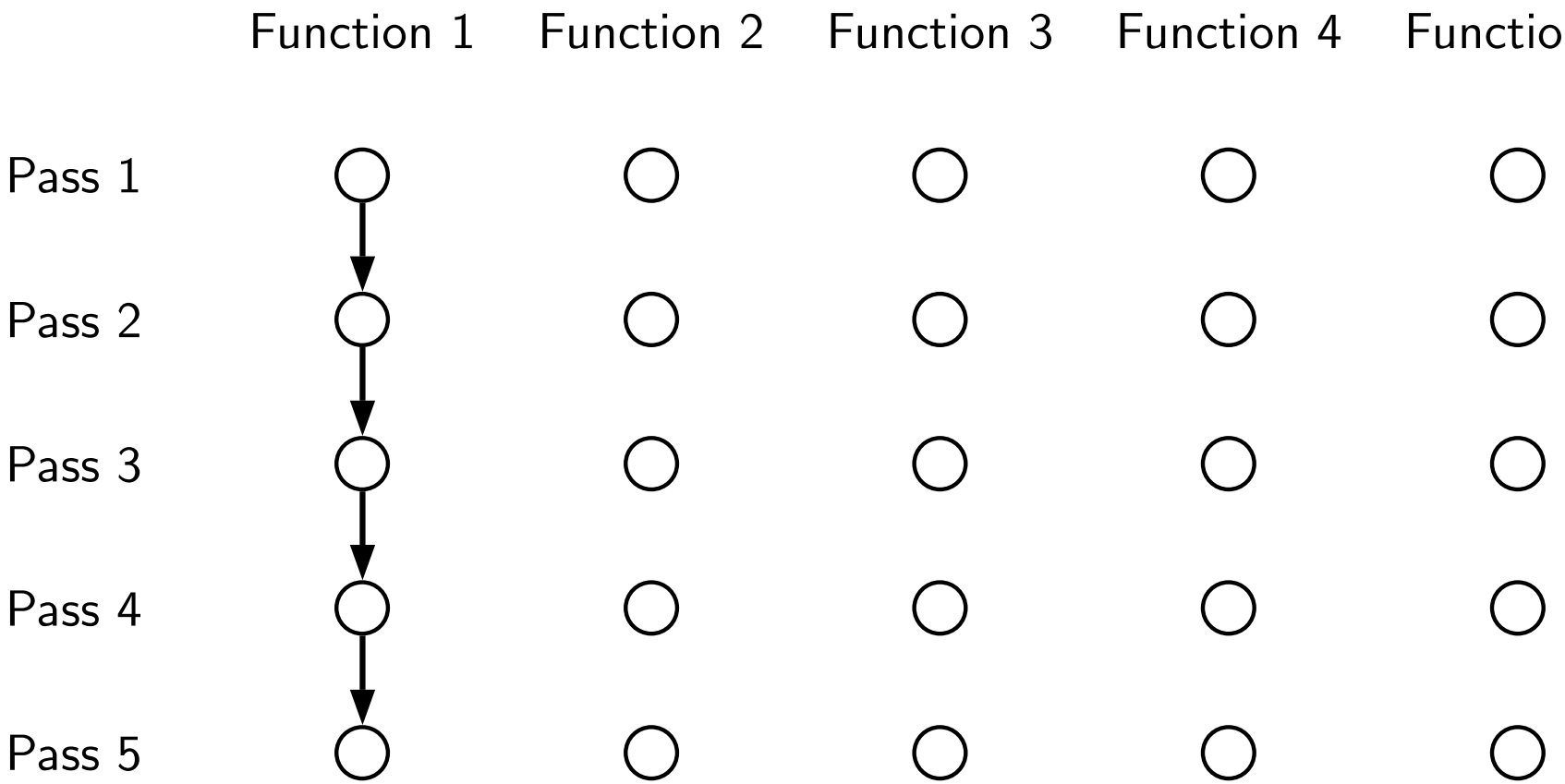
Execution Order in Intraprocedural Passes



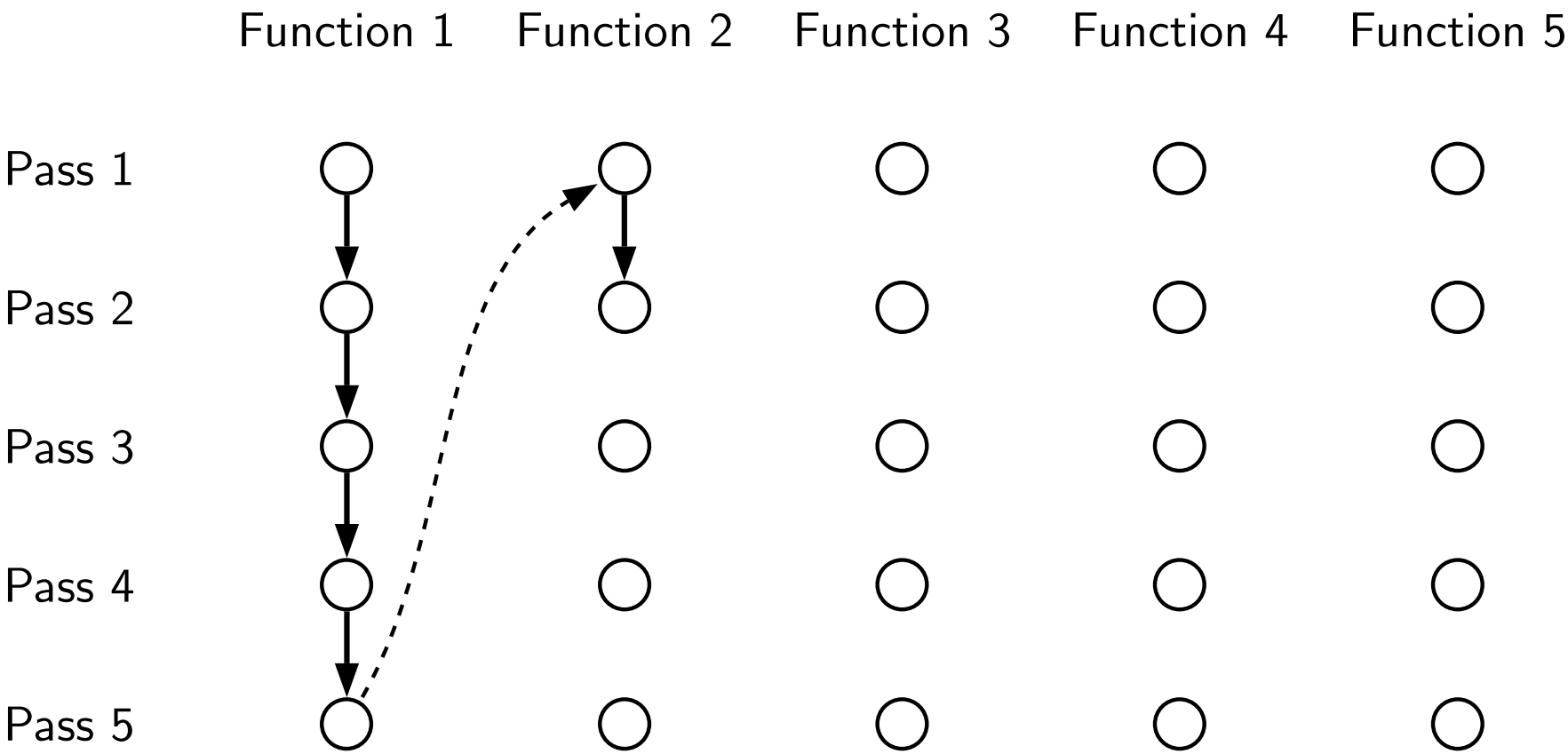
Execution Order in Intraprocedural Passes



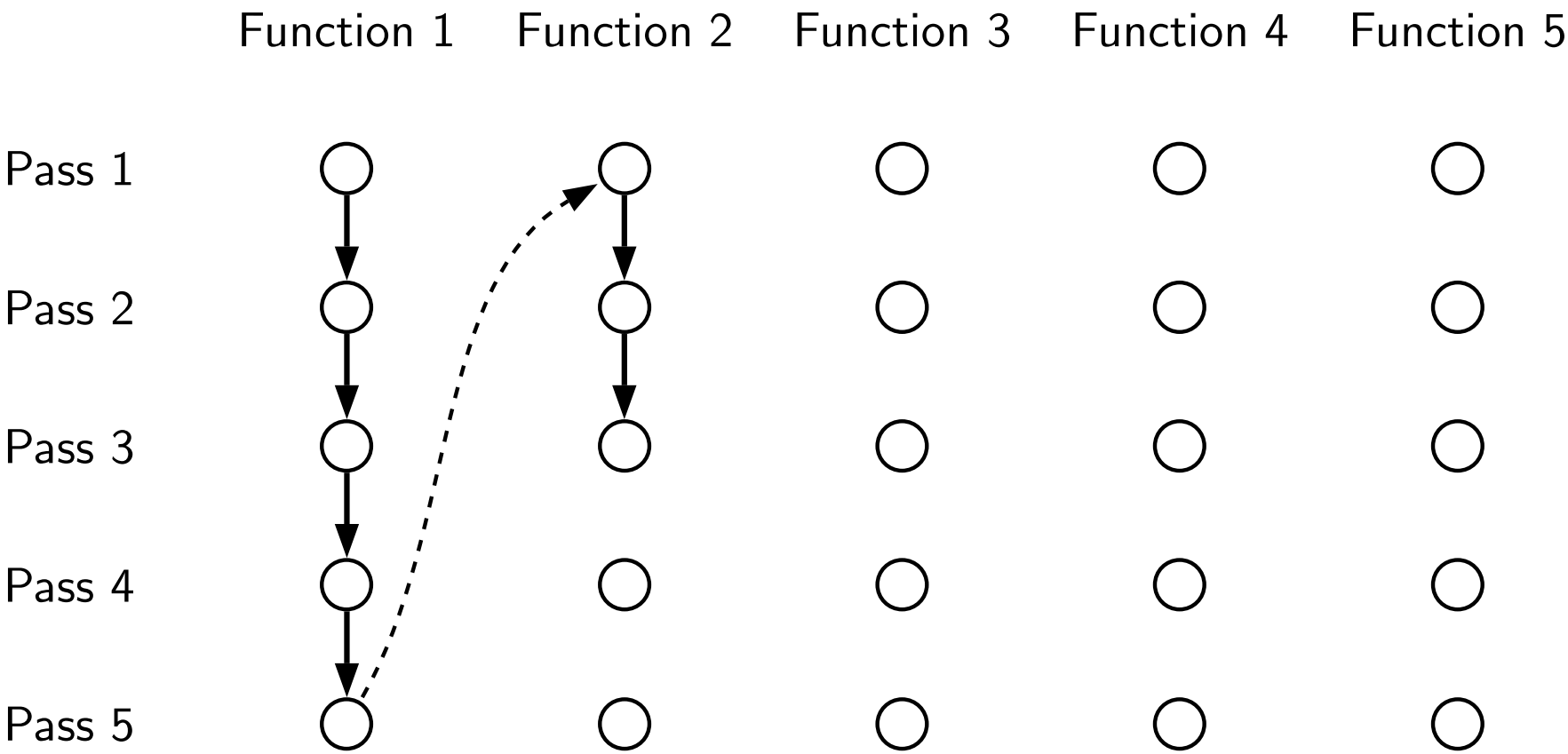
Execution Order in Intraprocedural Passes



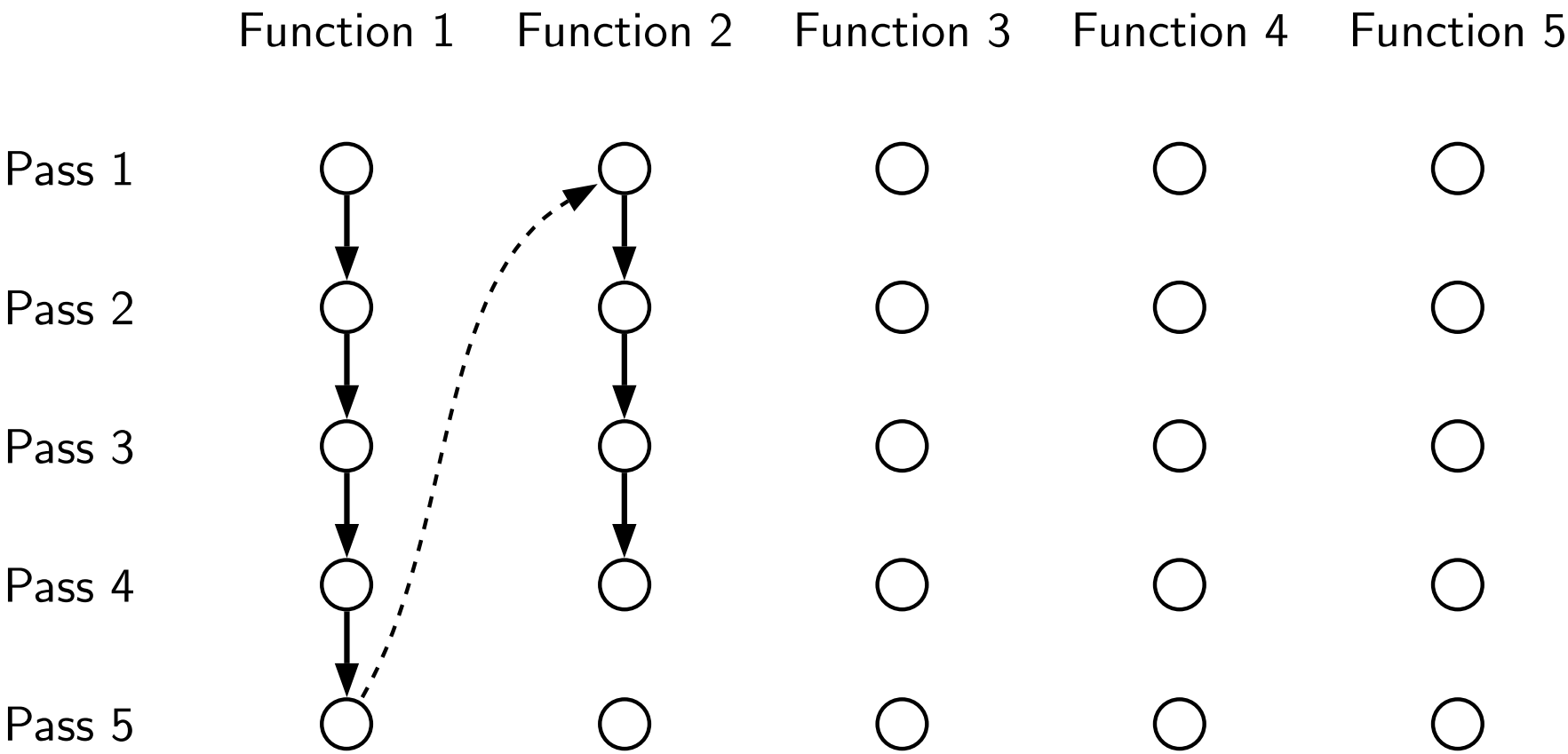
Execution Order in Intraprocedural Passes



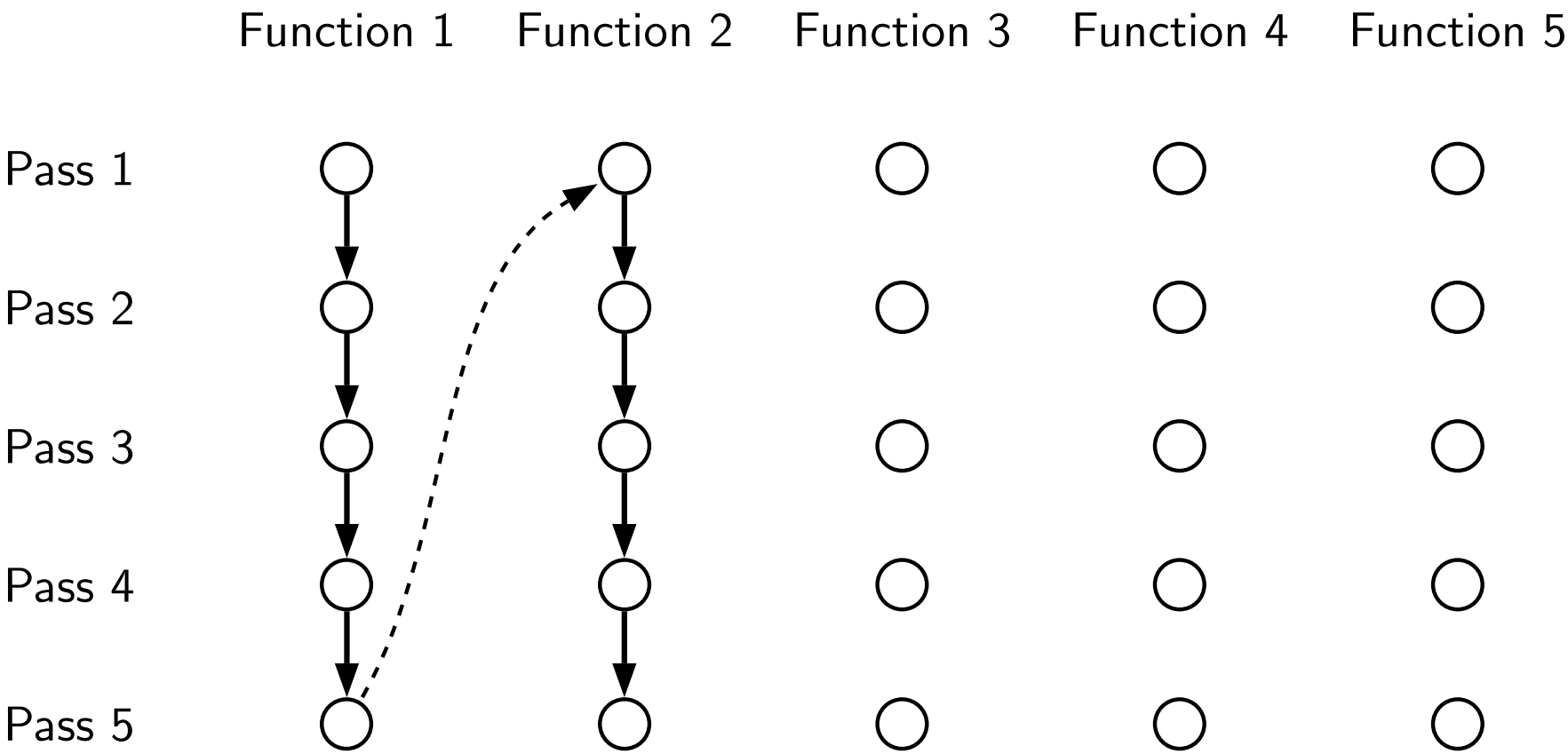
Execution Order in Intraprocedural Passes



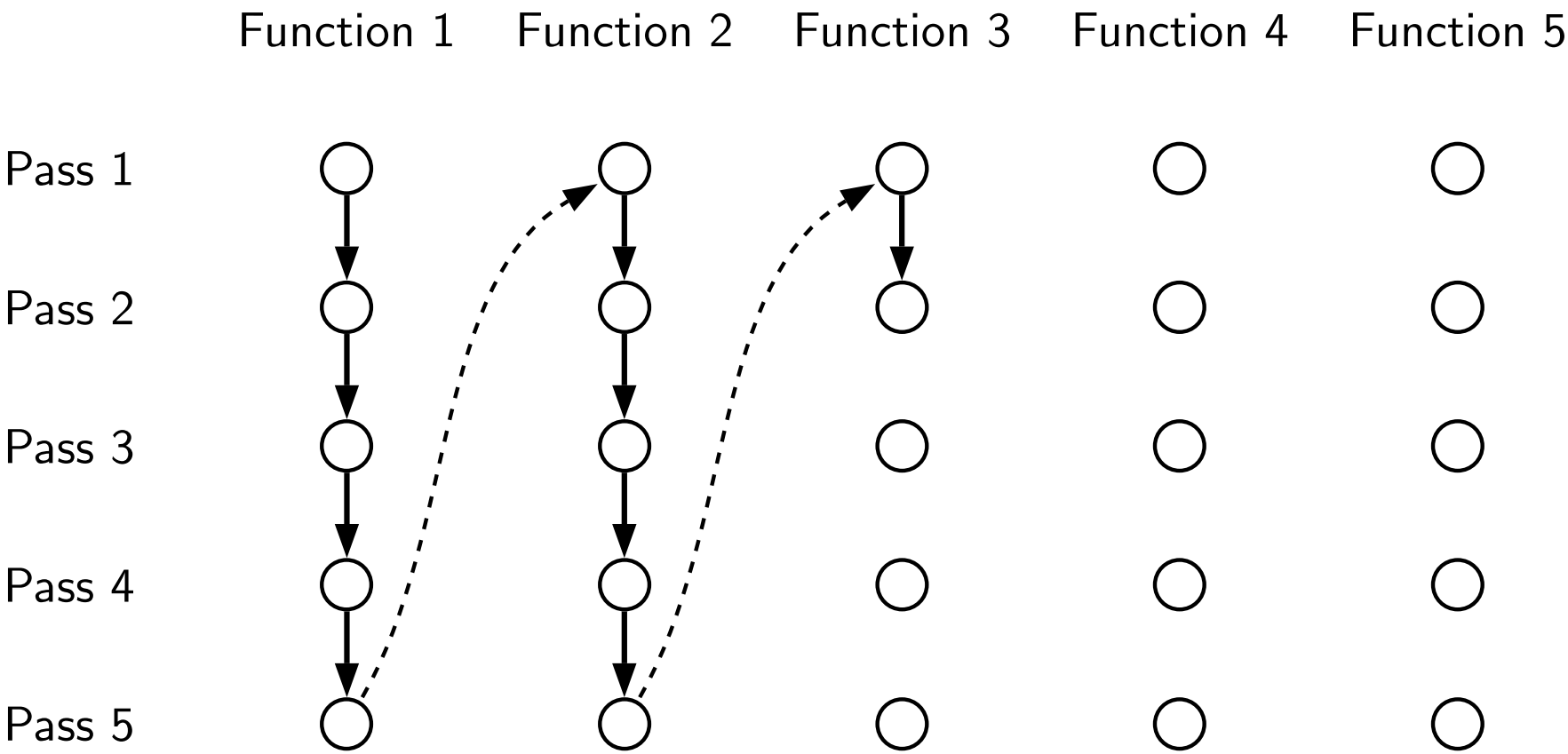
Execution Order in Intraprocedural Passes



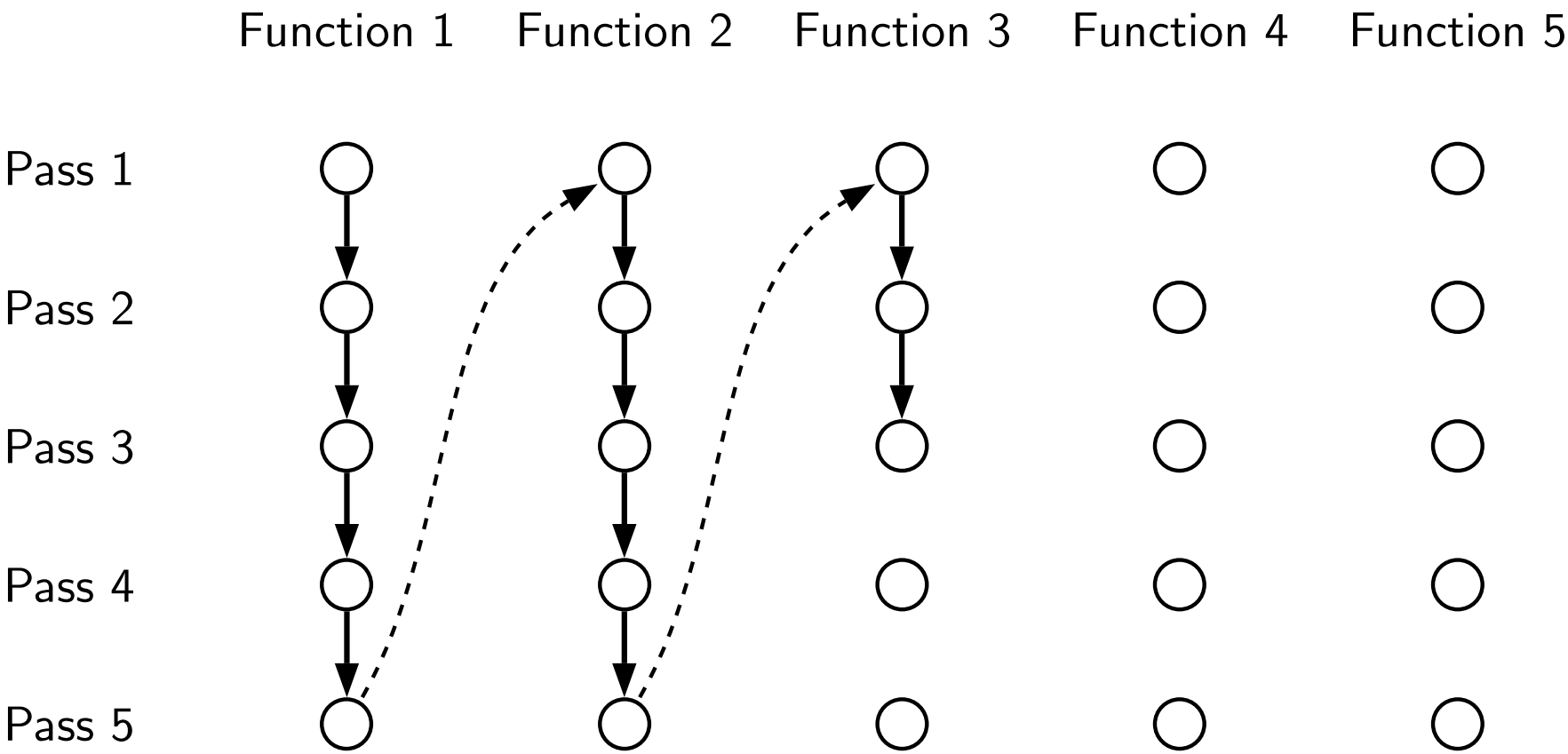
Execution Order in Intraprocedural Passes



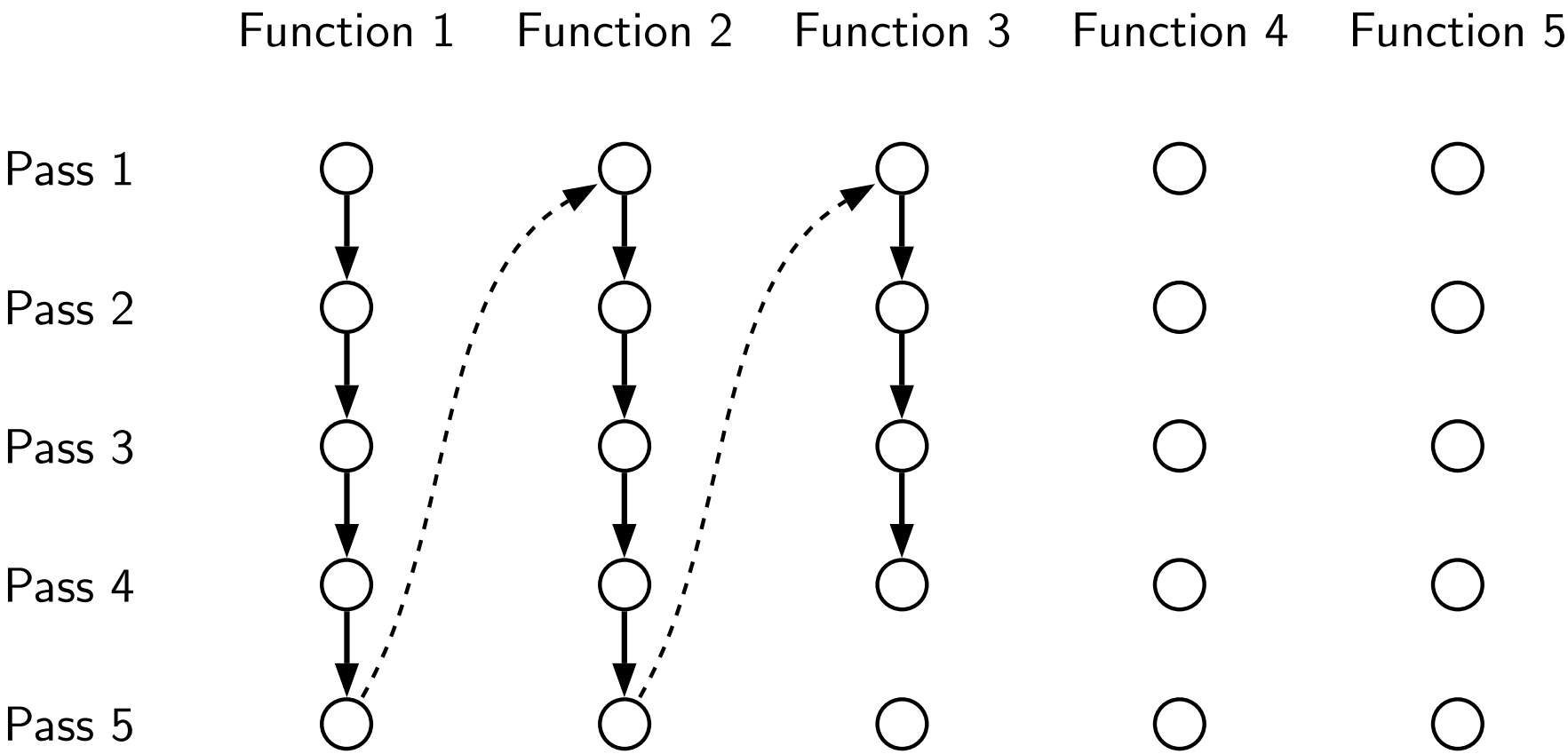
Execution Order in Intraprocedural Passes



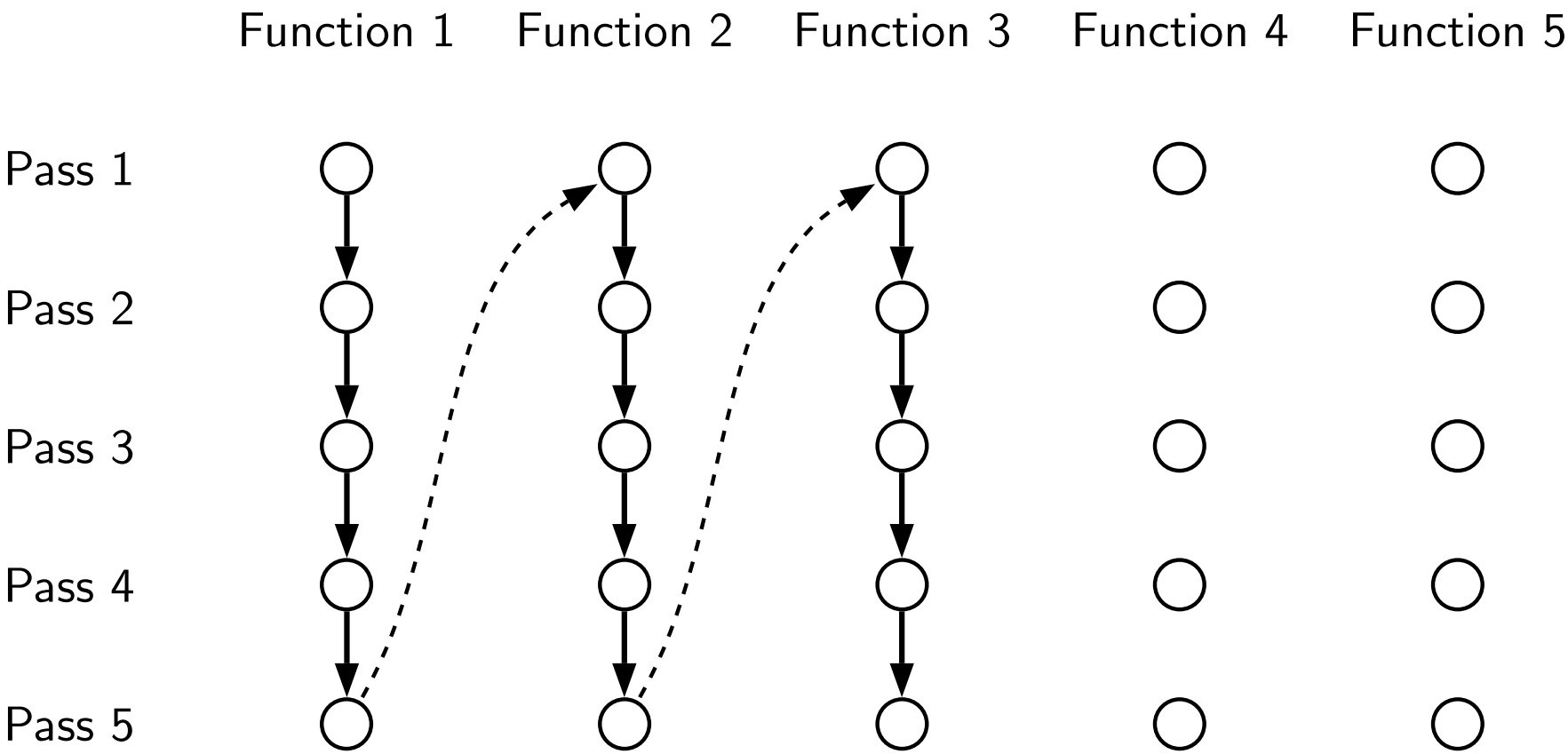
Execution Order in Intraprocedural Passes



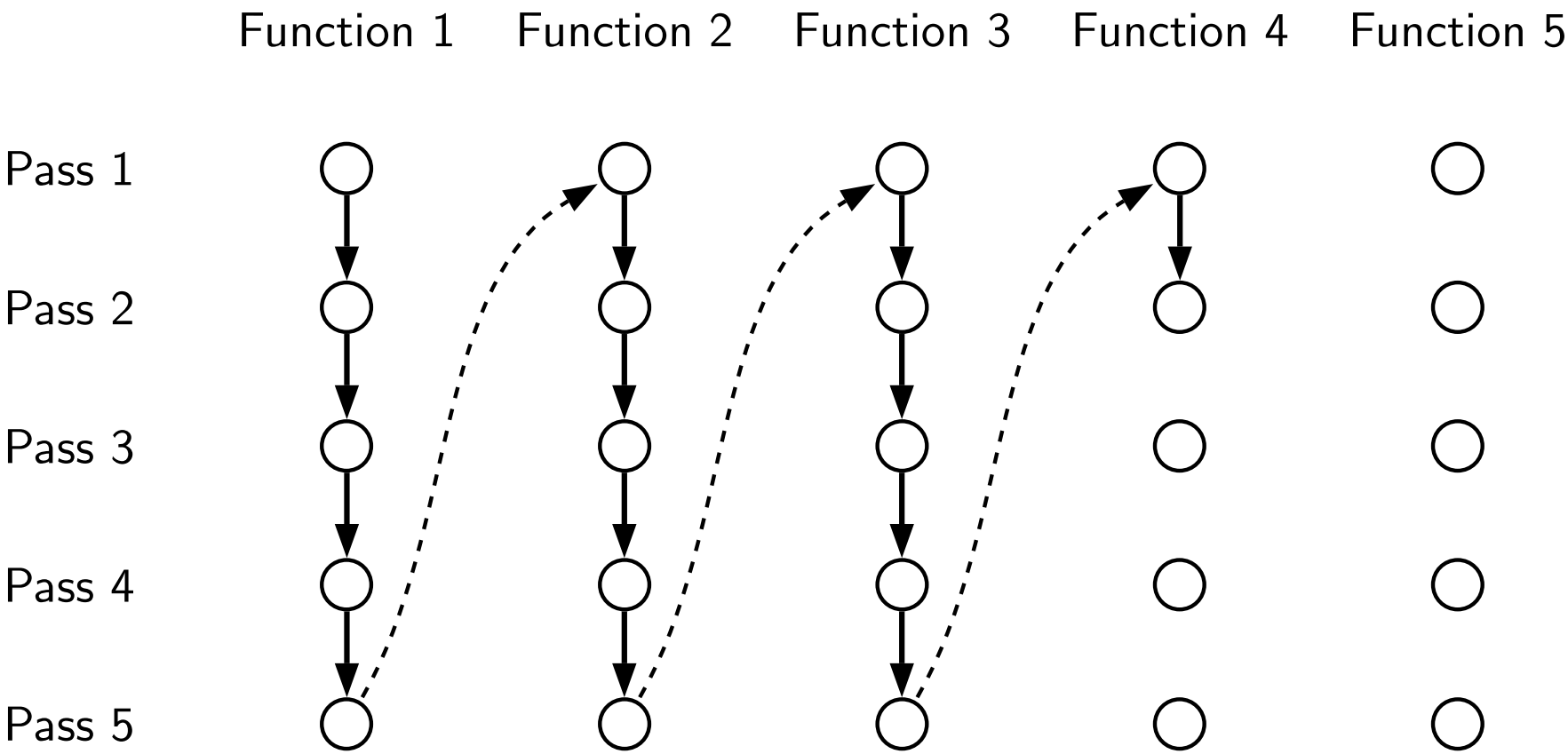
Execution Order in Intraprocedural Passes



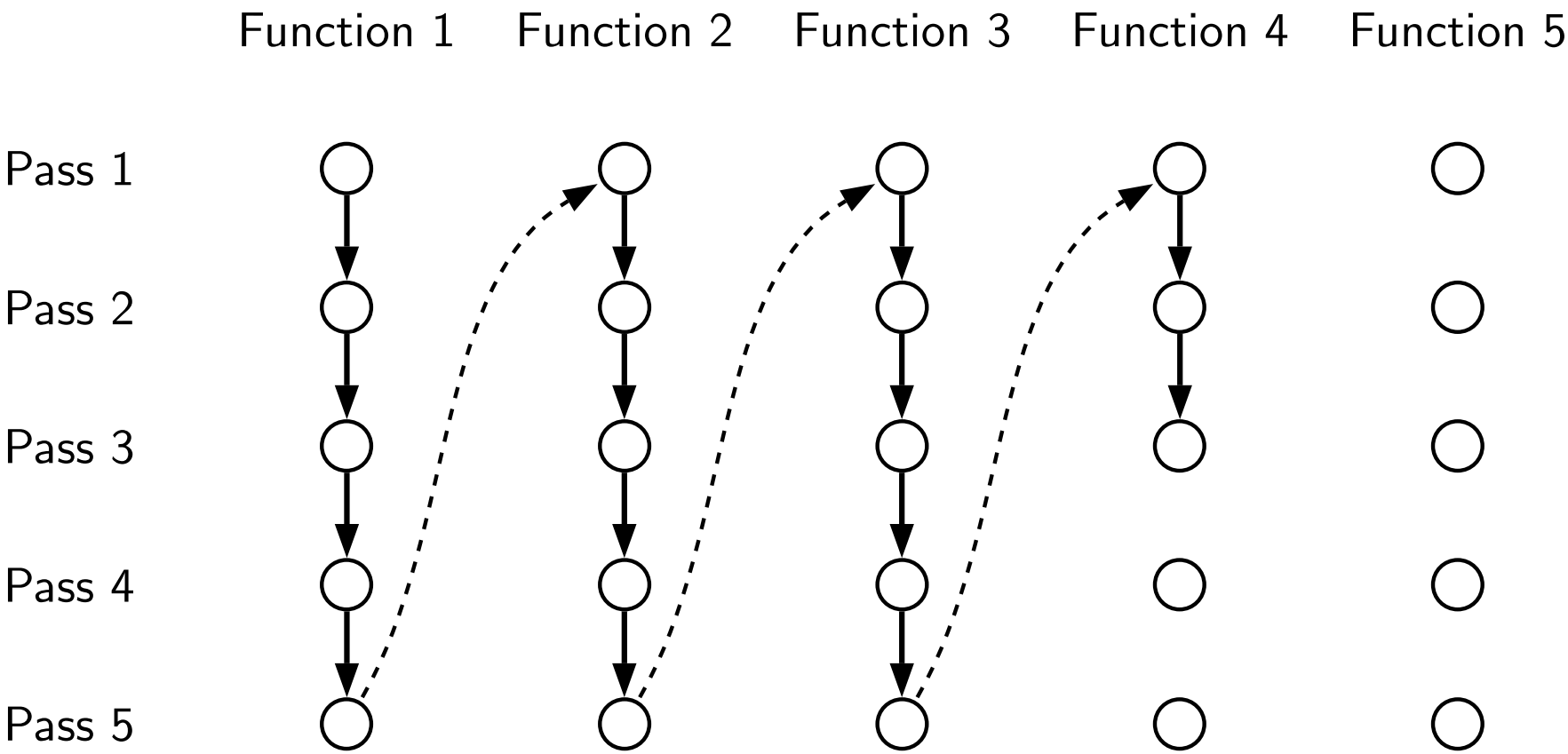
Execution Order in Interprocedural Passes



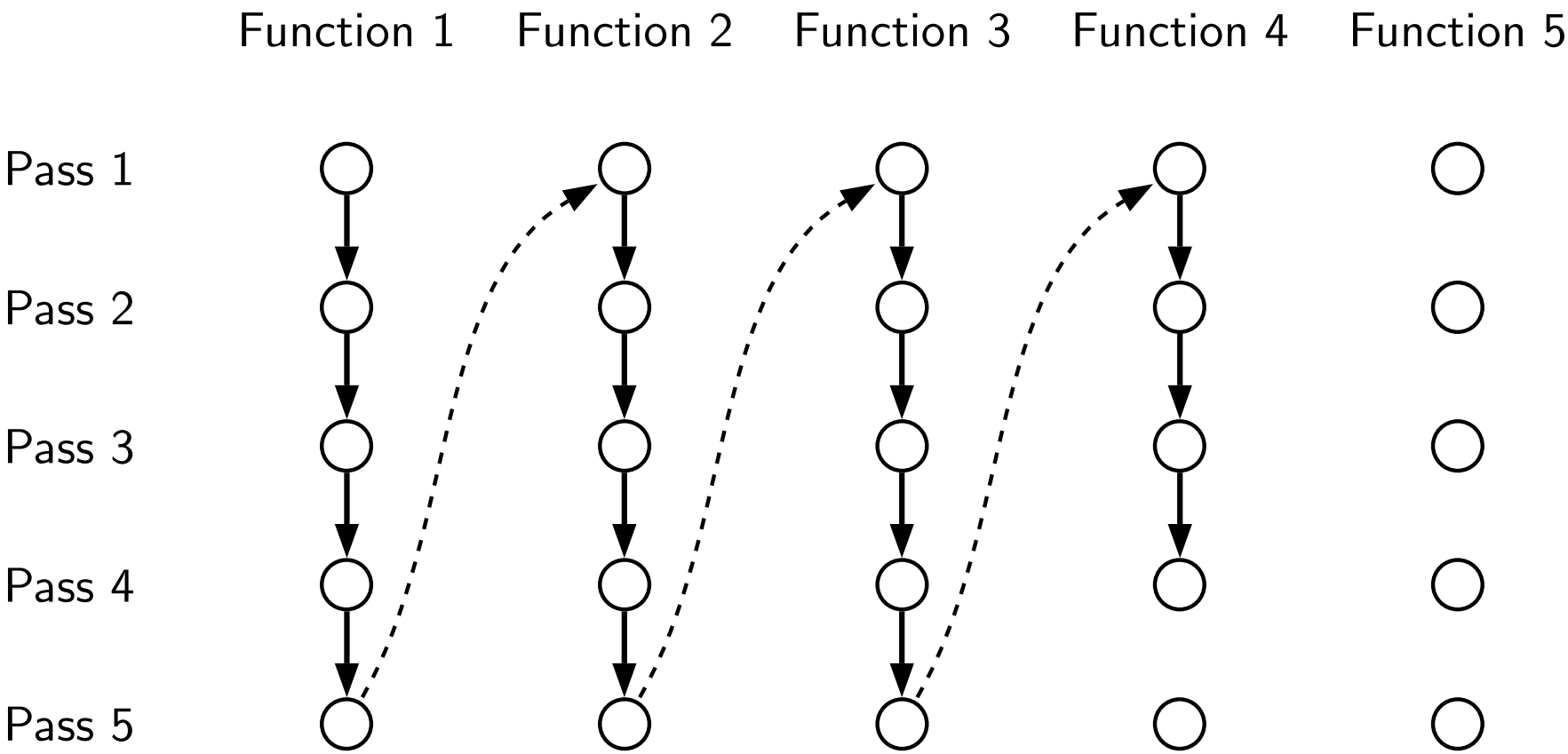
Execution Order in Interprocedural Passes



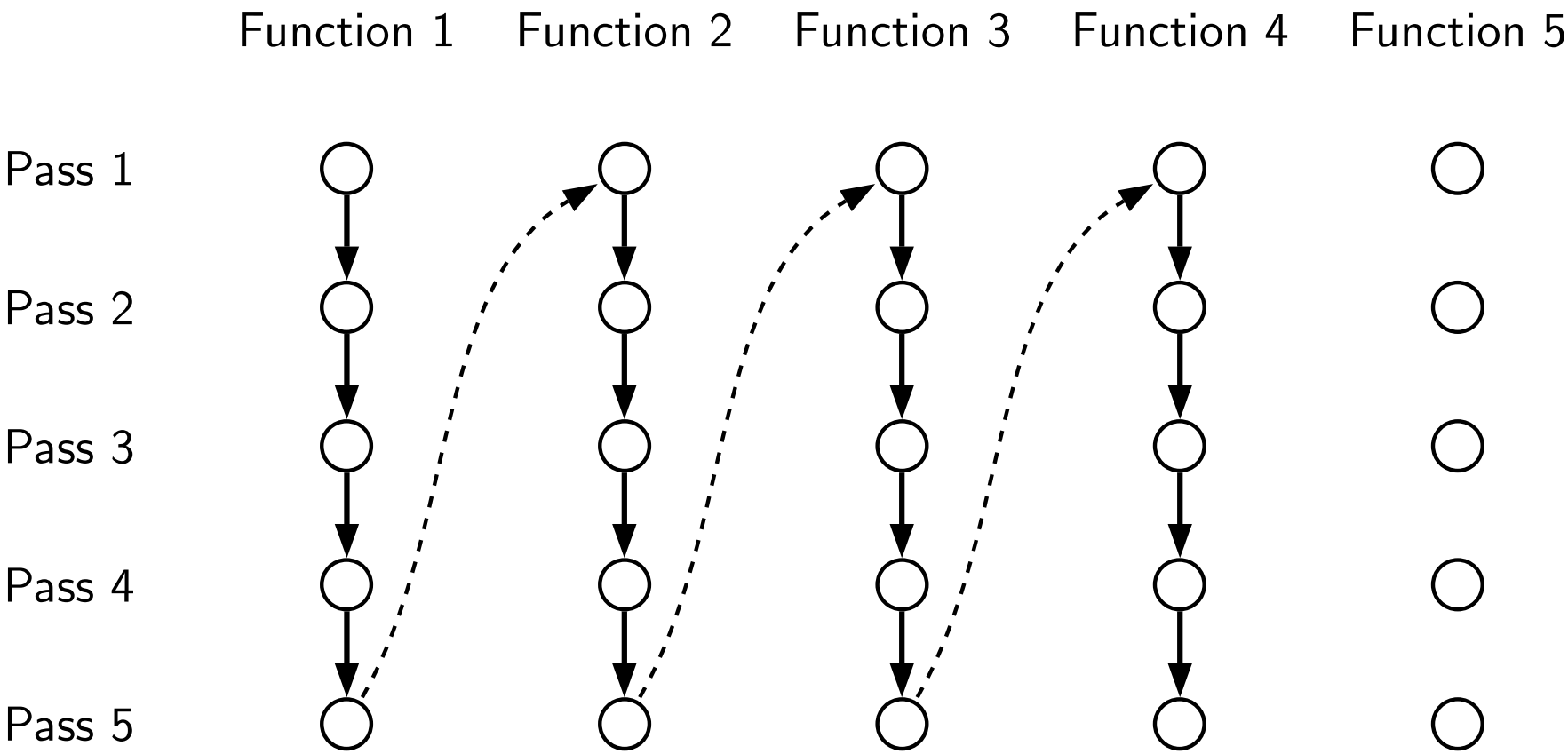
Execution Order in Intraprocedural Passes



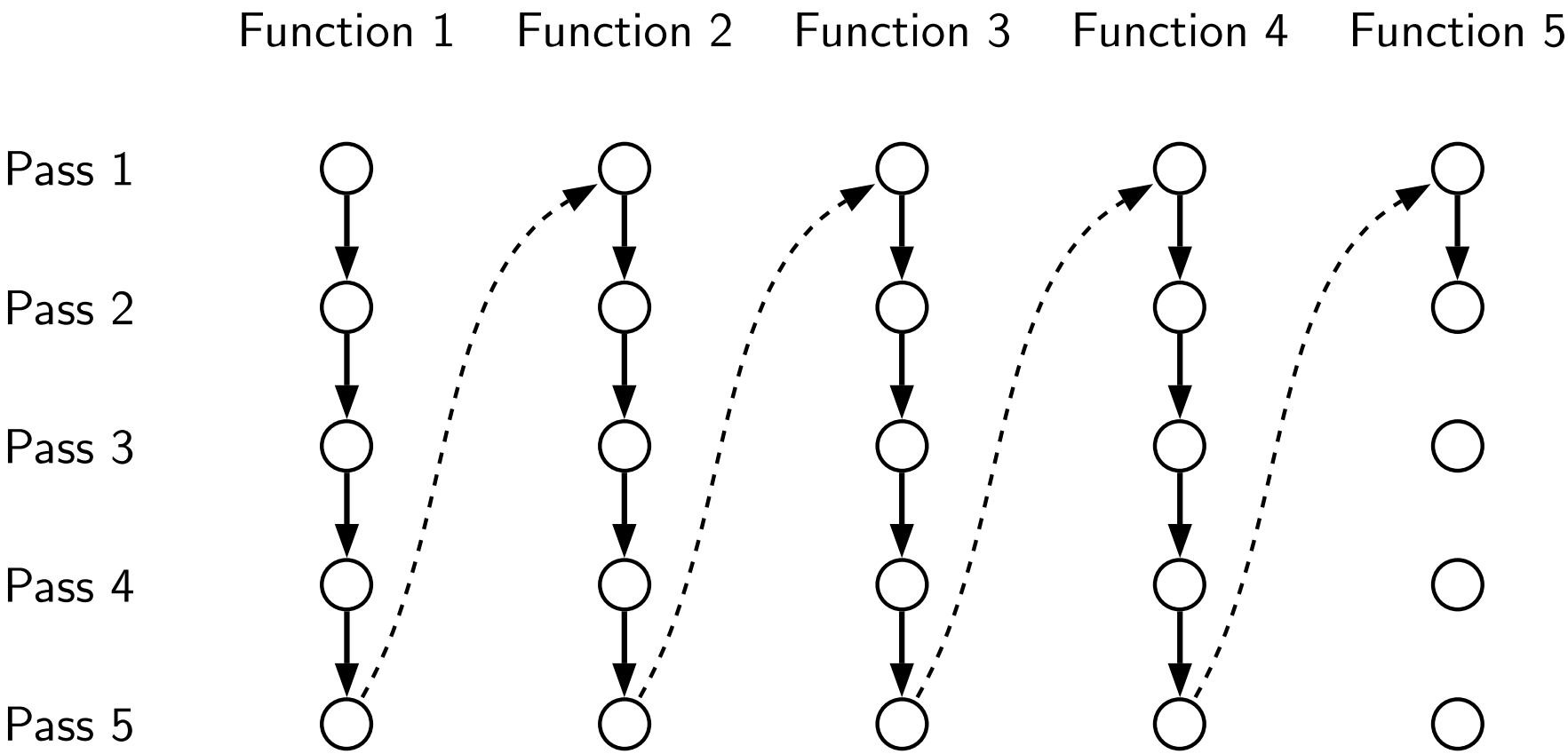
Execution Order in Intraprocedural Passes



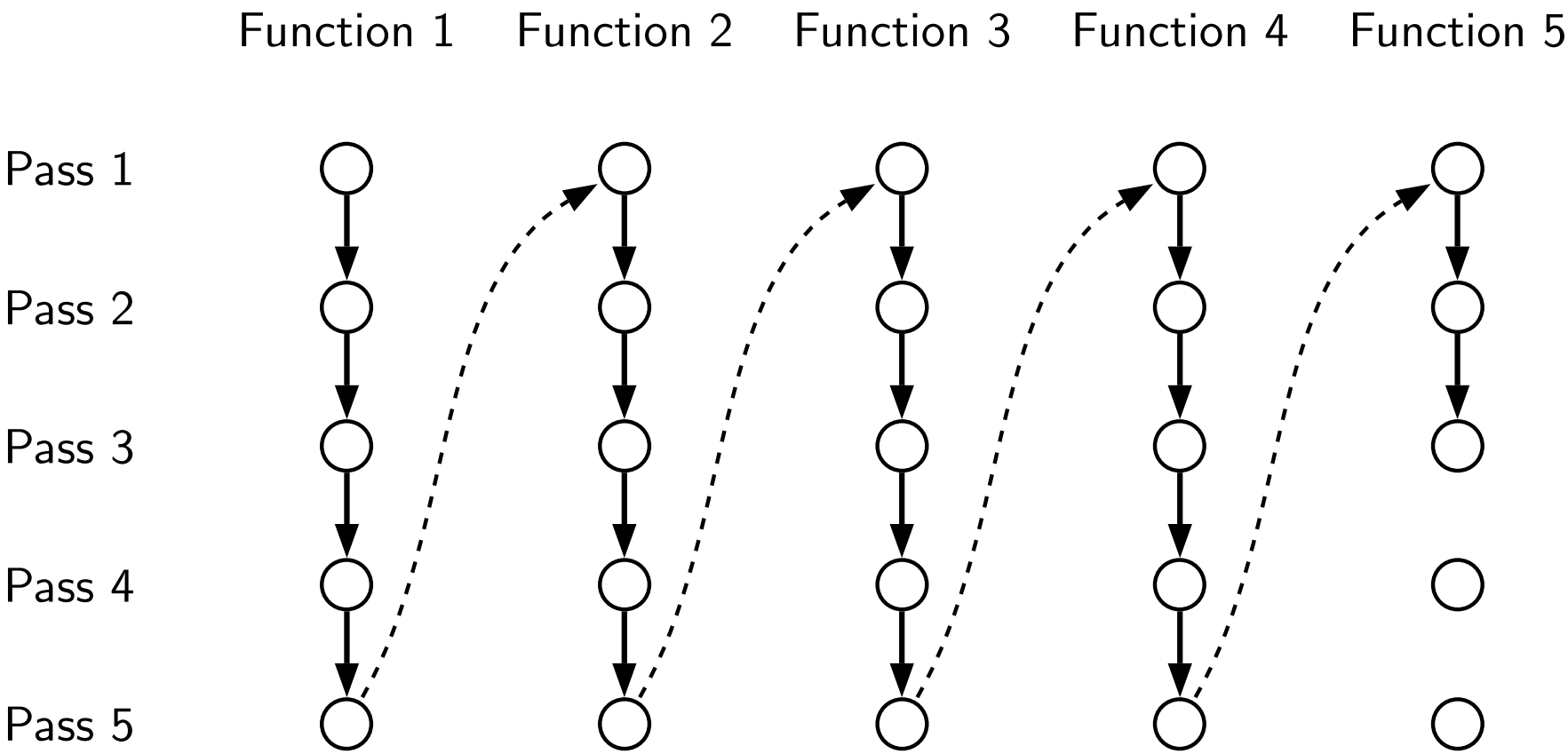
Execution Order in Intraprocedural Passes



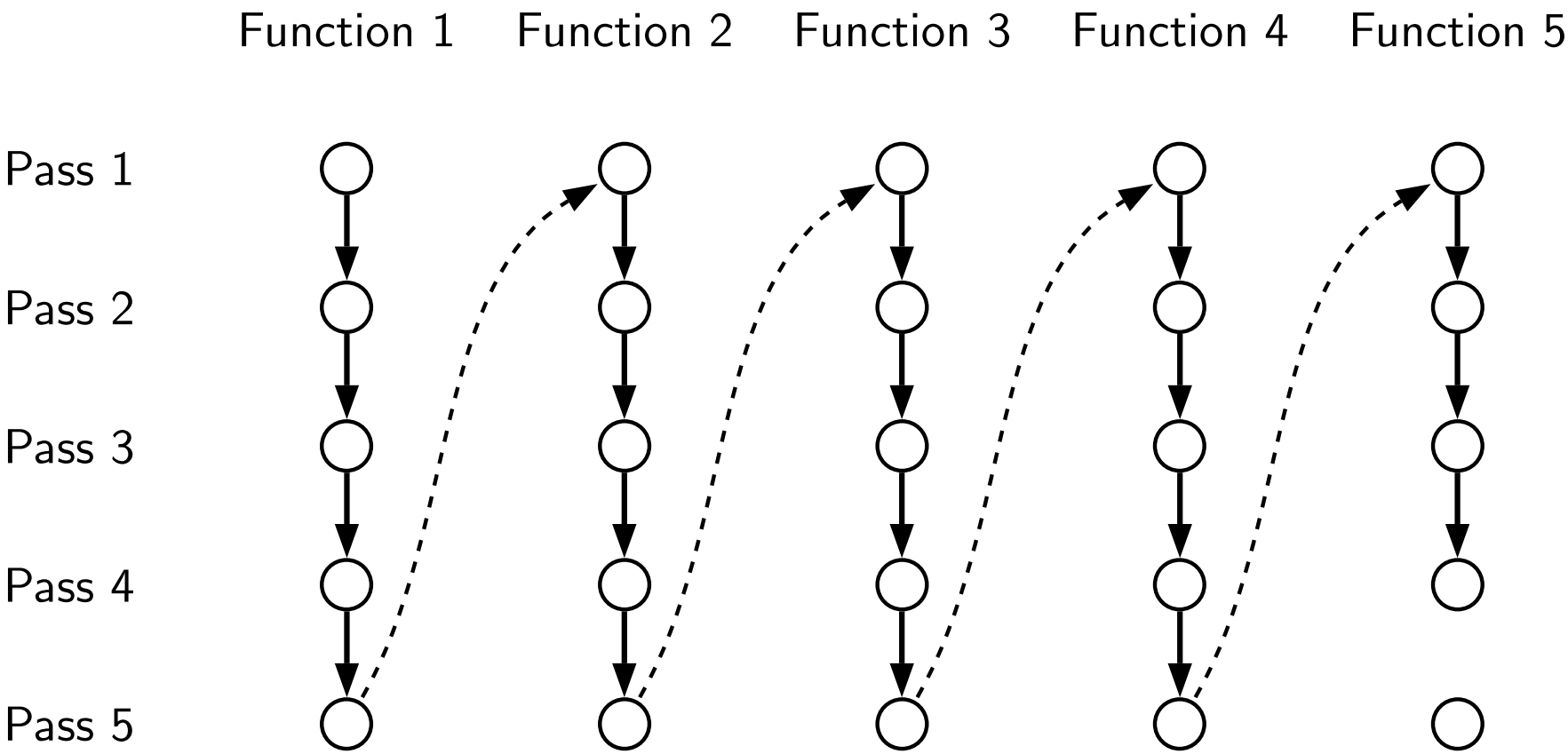
Execution Order in Intraprocedural Passes



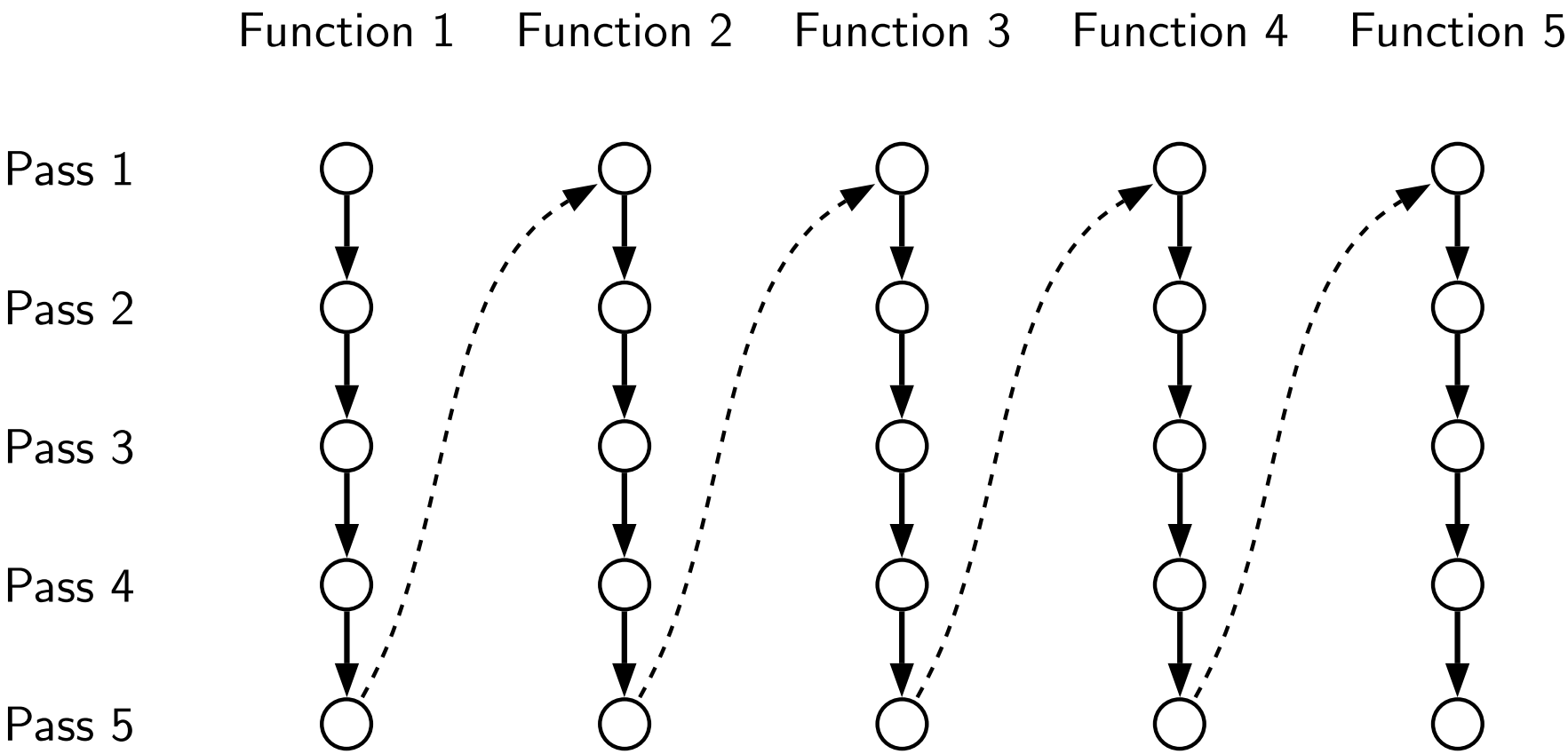
Execution Order in Intraprocedural Passes



Execution Order in Intraprocedural Passes



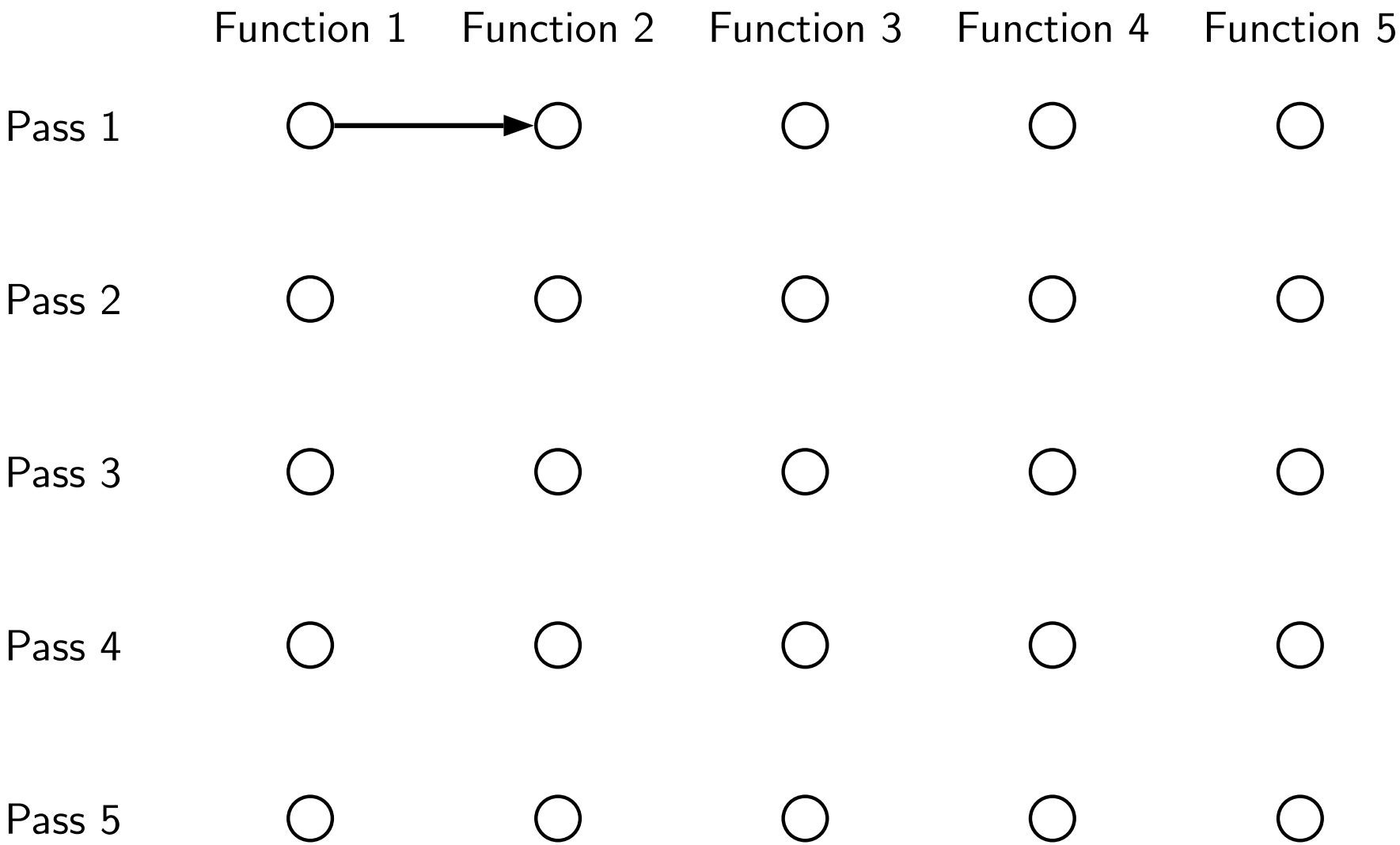
Execution Order in Intraprocedural Passes



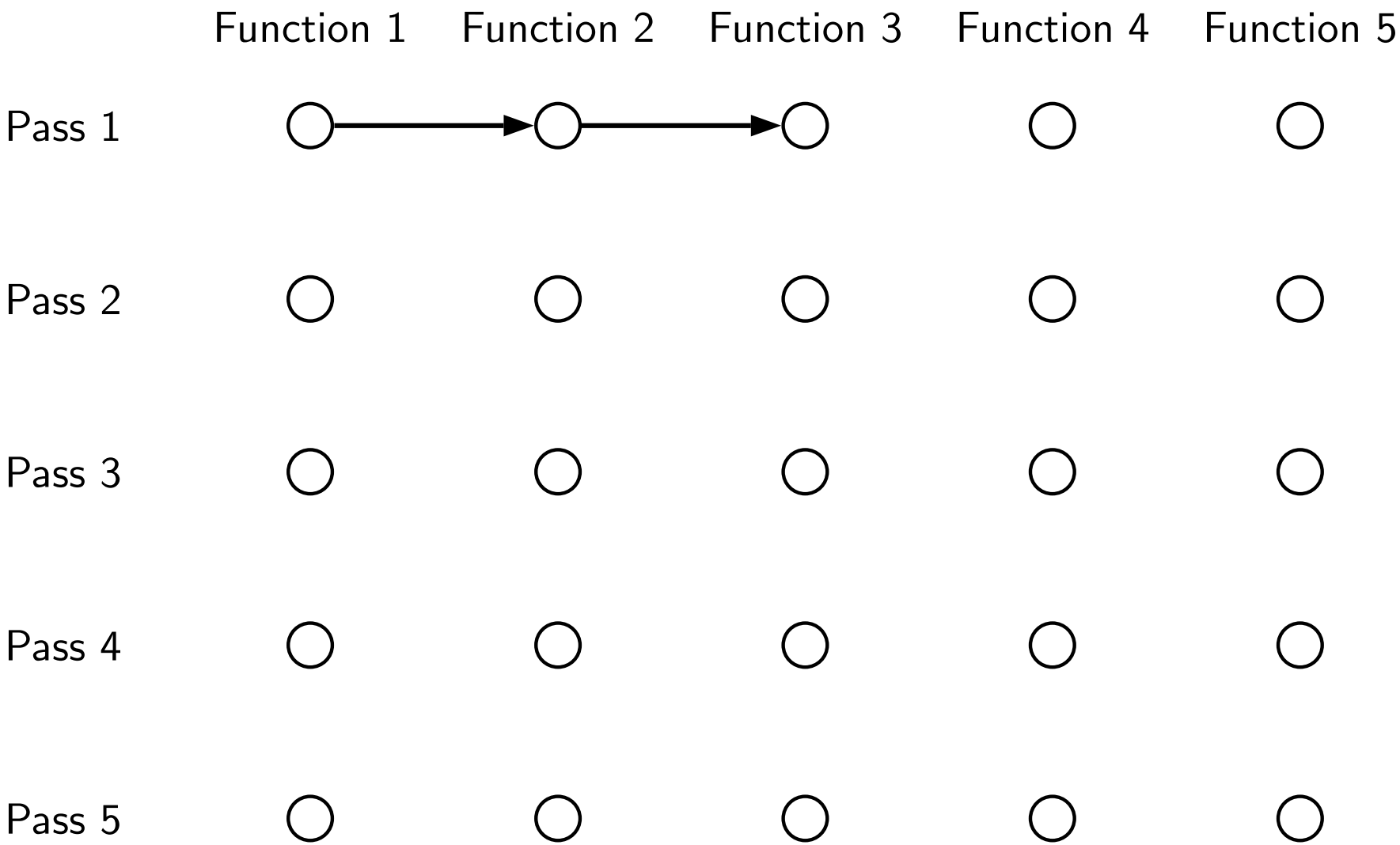
Execution Order in Interprocedural Passes

	Function 1	Function 2	Function 3	Function 4	Function 5
Pass 1	○	○	○	○	○
Pass 2	○	○	○	○	○
Pass 3	○	○	○	○	○
Pass 4	○	○	○	○	○
Pass 5	○	○	○	○	○

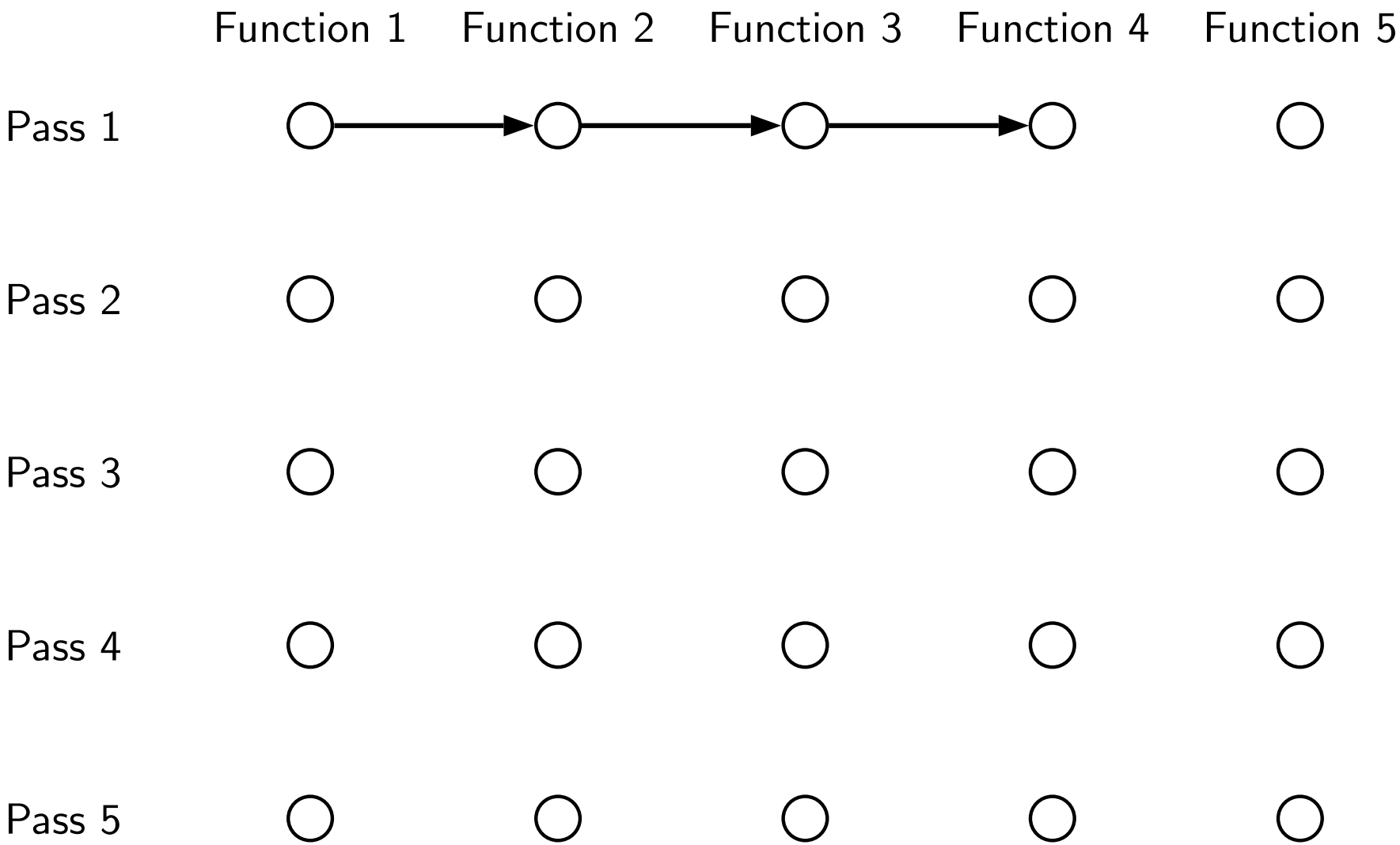
Execution Order in Interprocedural Passes



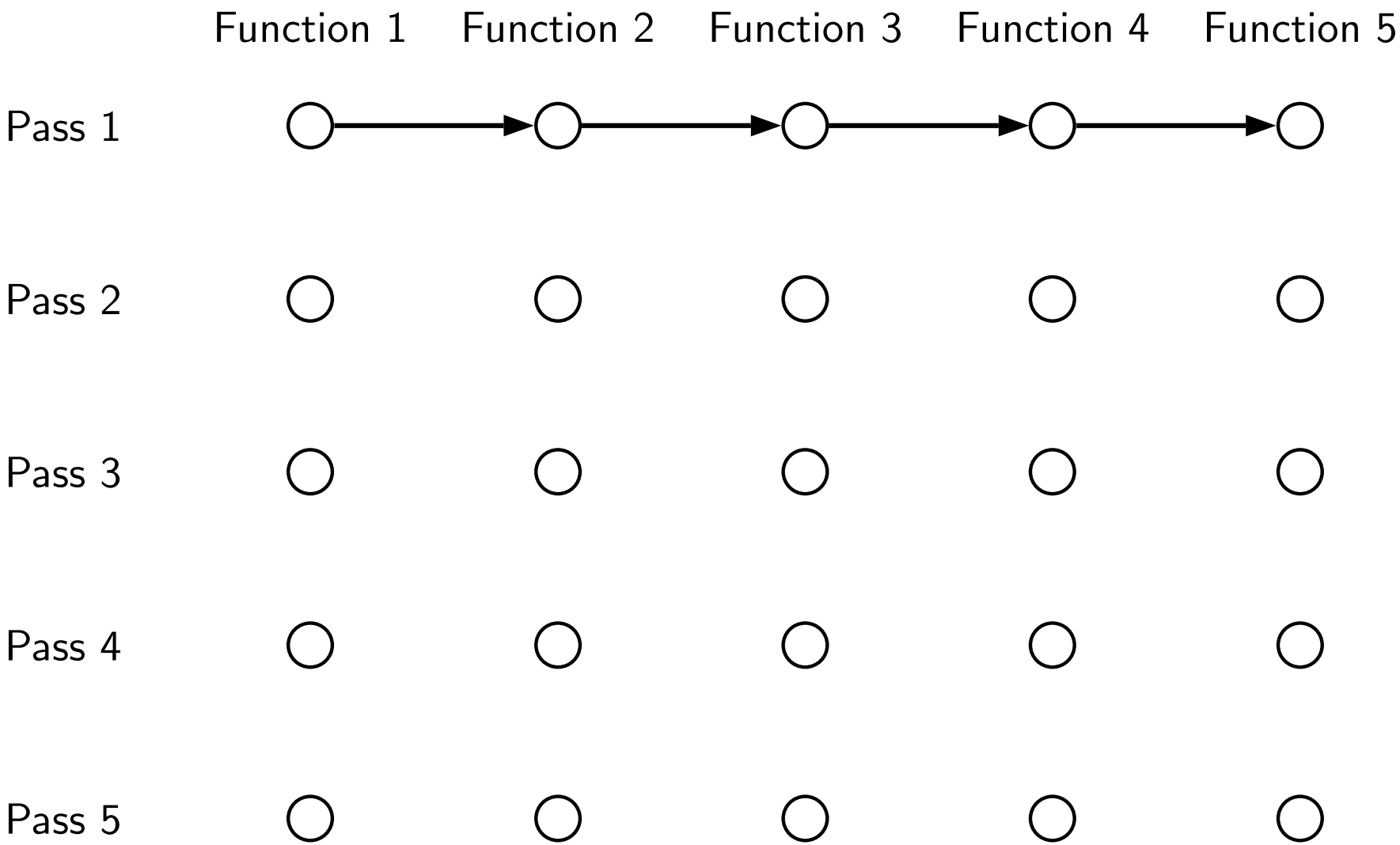
Execution Order in Interprocedural Passes



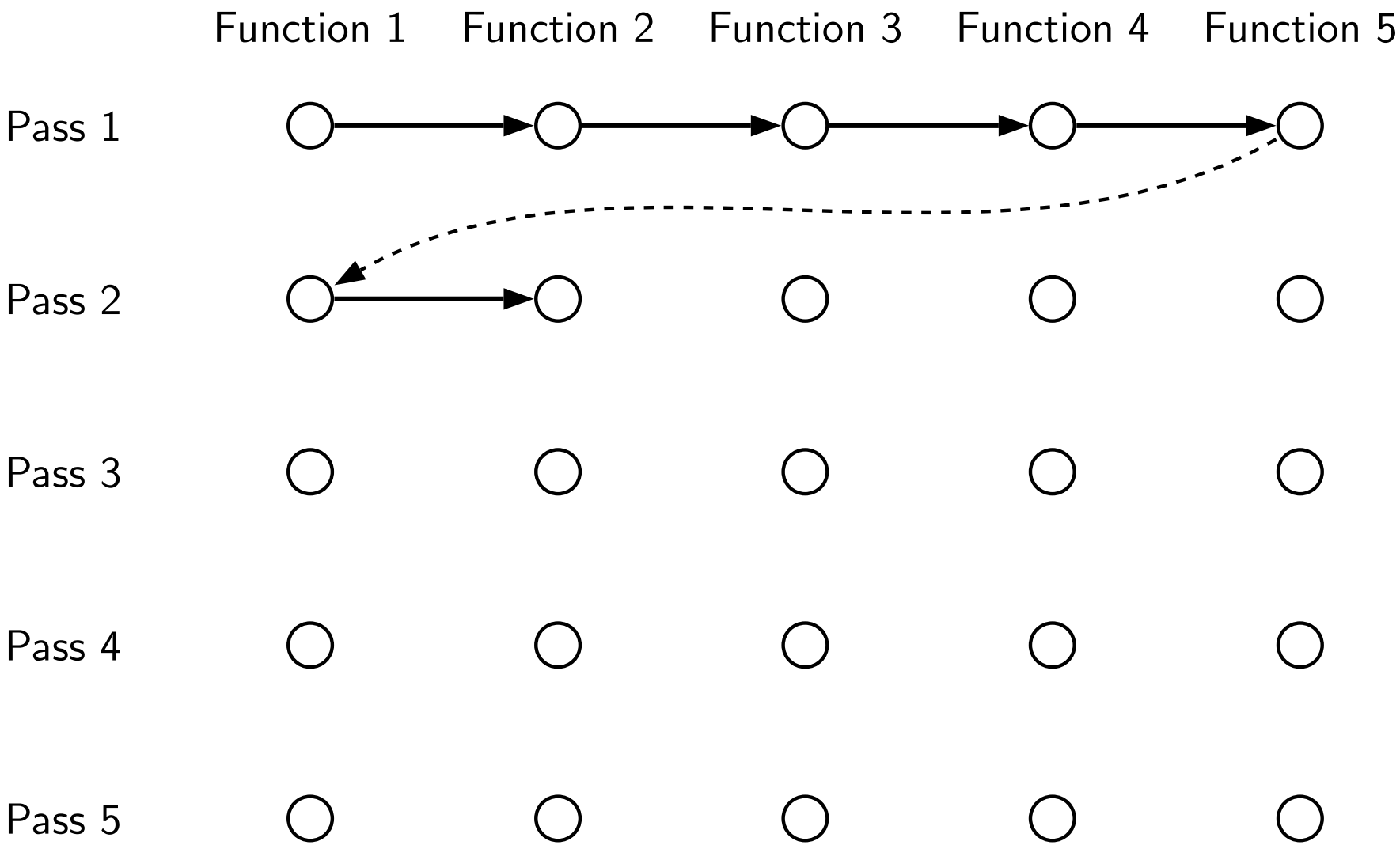
Execution Order in Interprocedural Passes



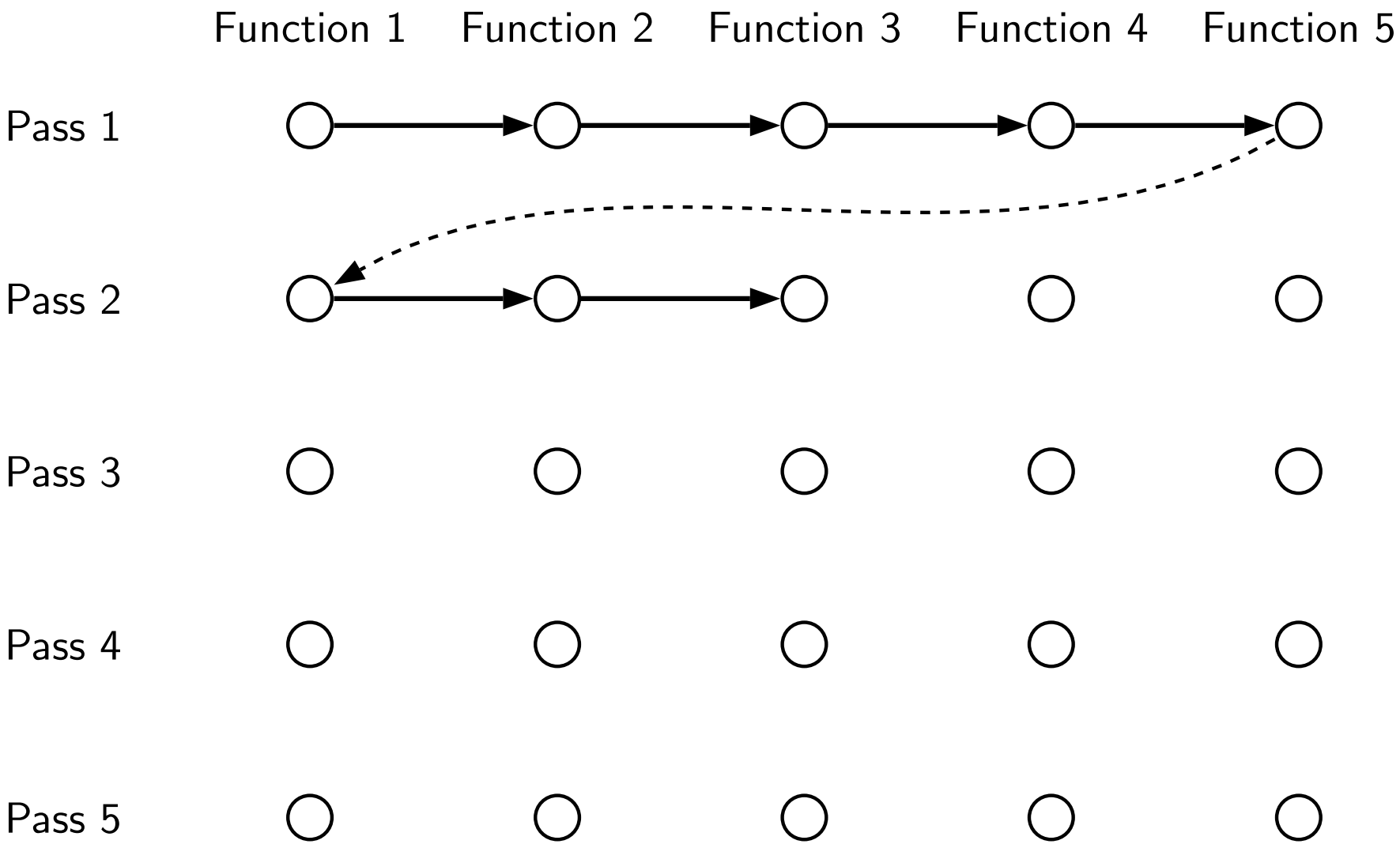
Execution Order in Interprocedural Passes



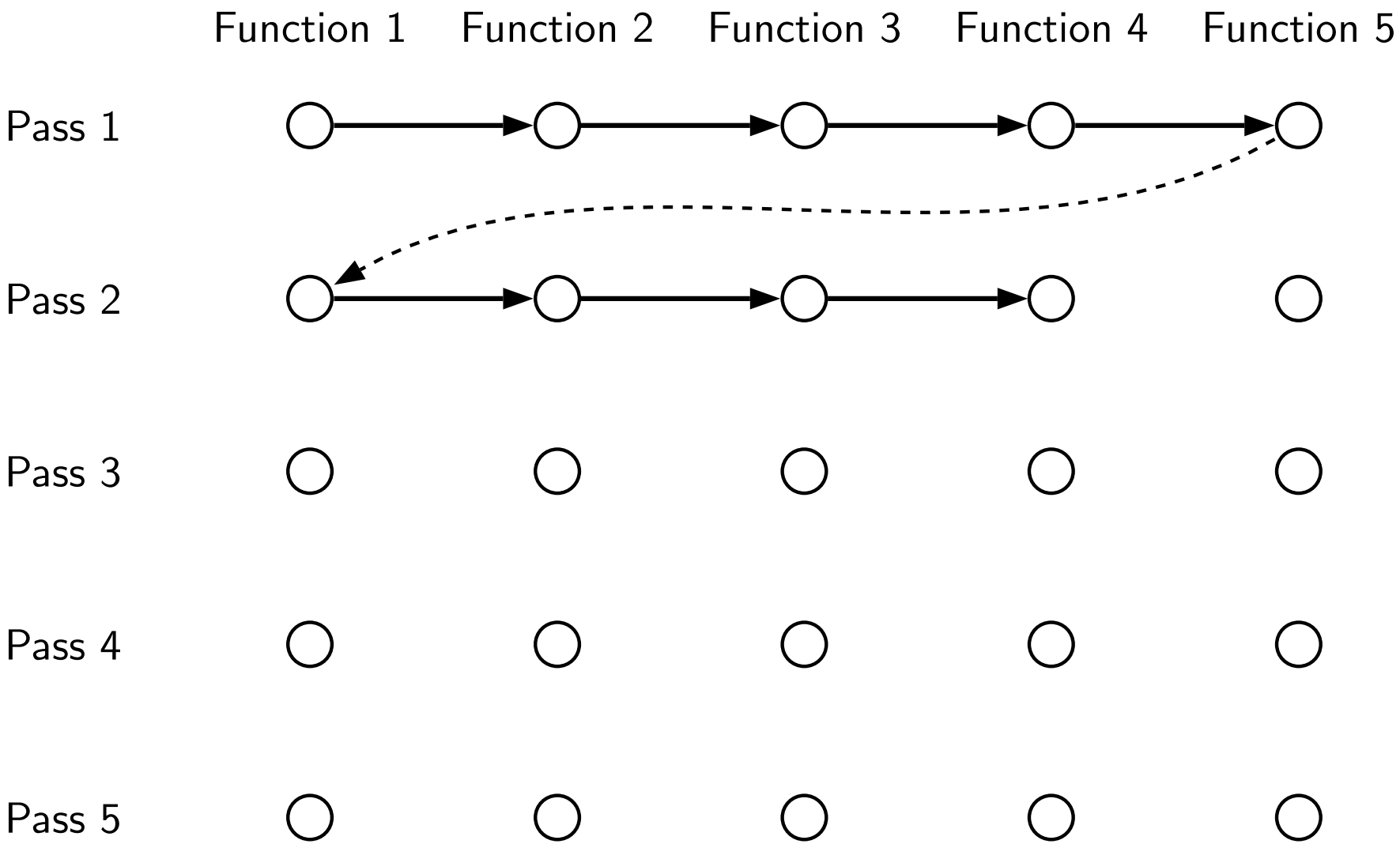
Execution Order in Interprocedural Passes



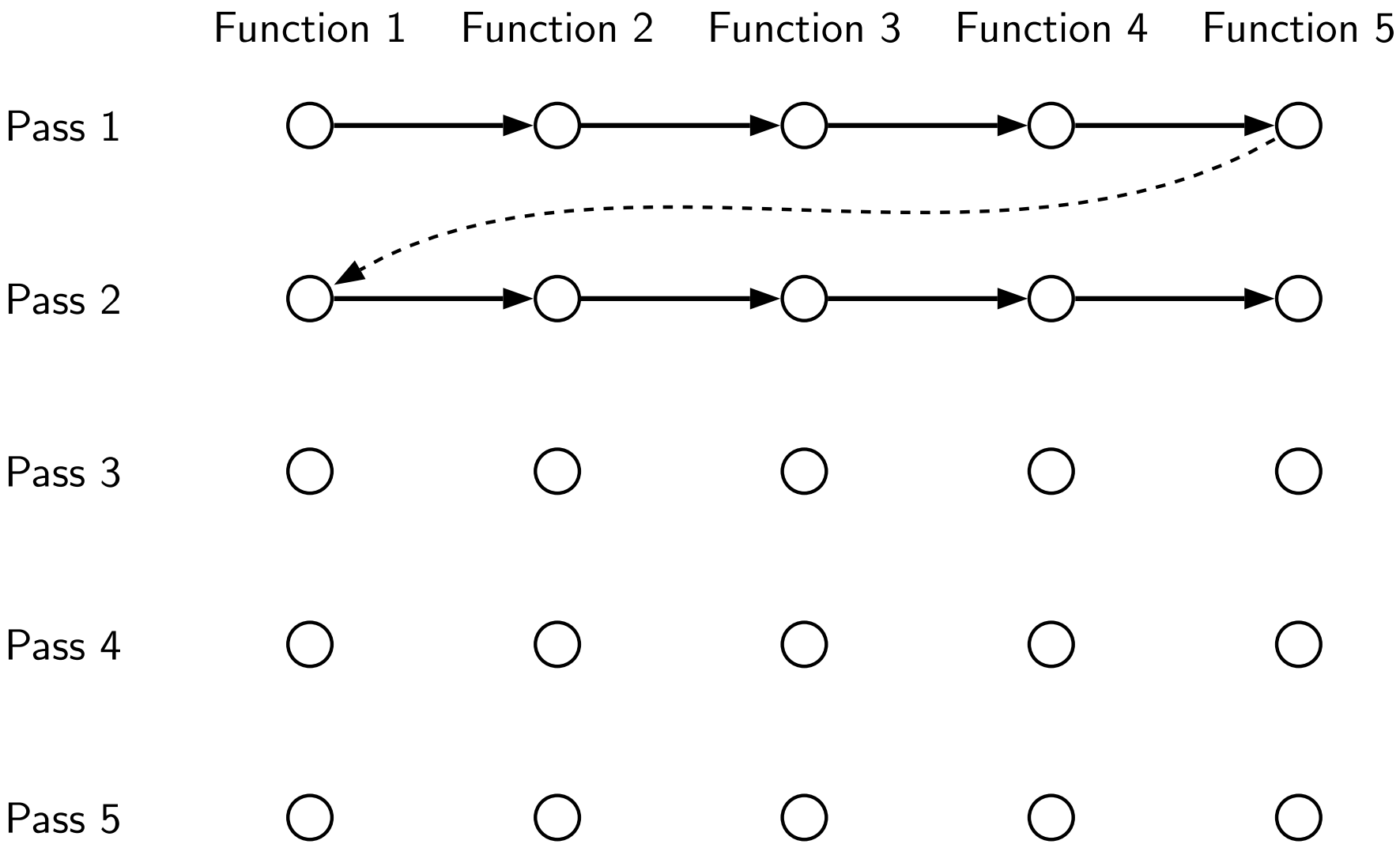
Execution Order in Interprocedural Passes



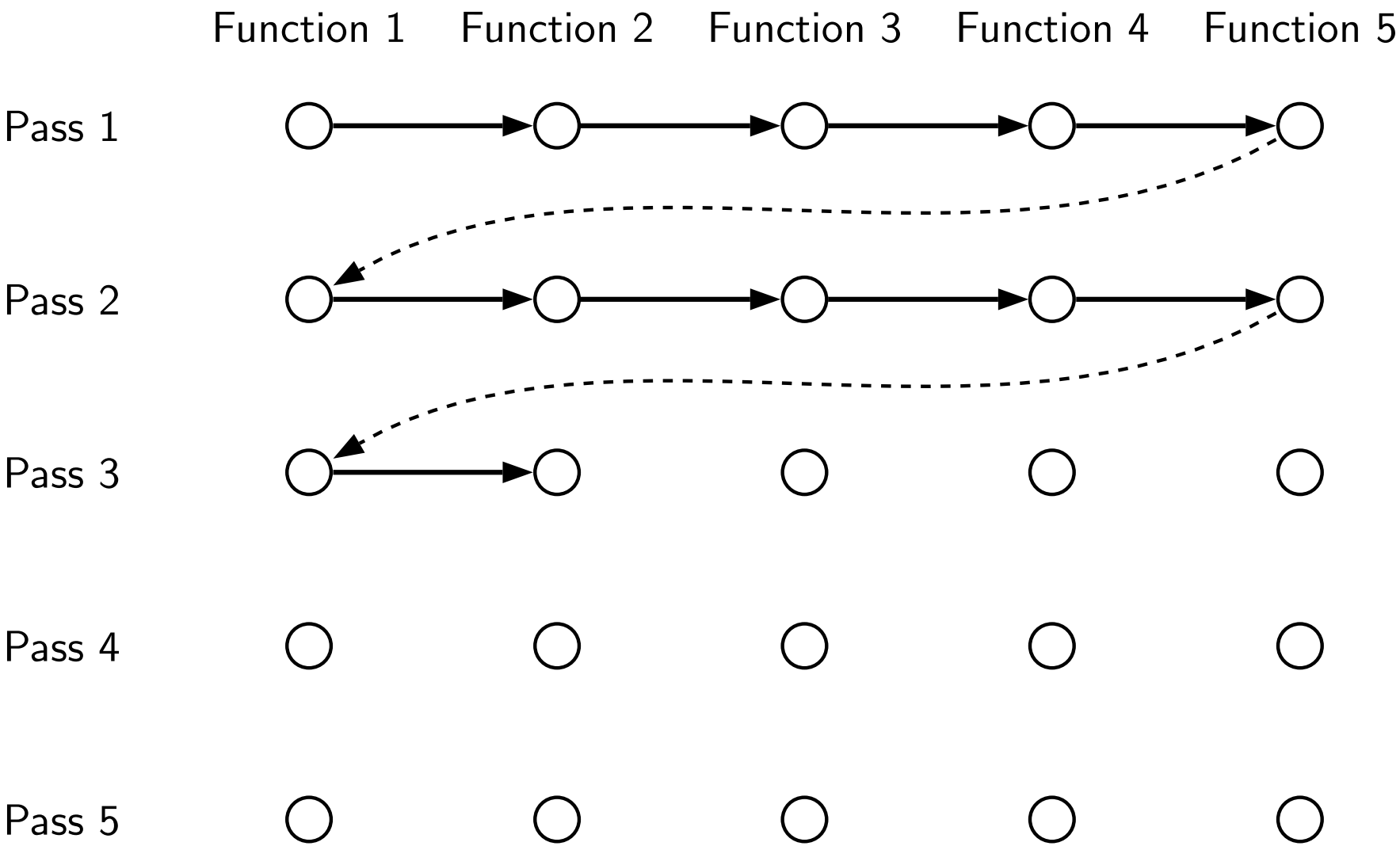
Execution Order in Interprocedural Passes



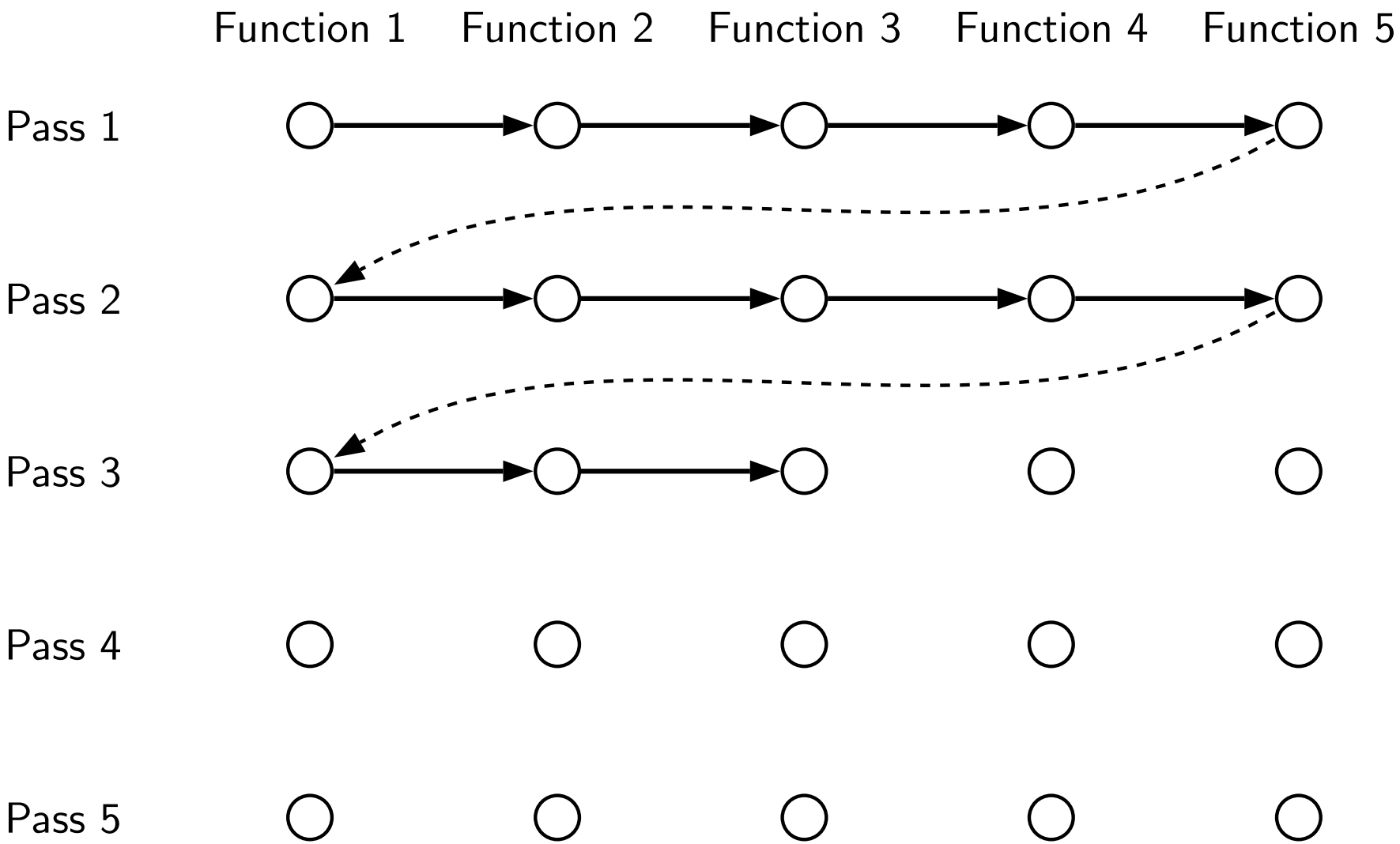
Execution Order in Interprocedural Passes



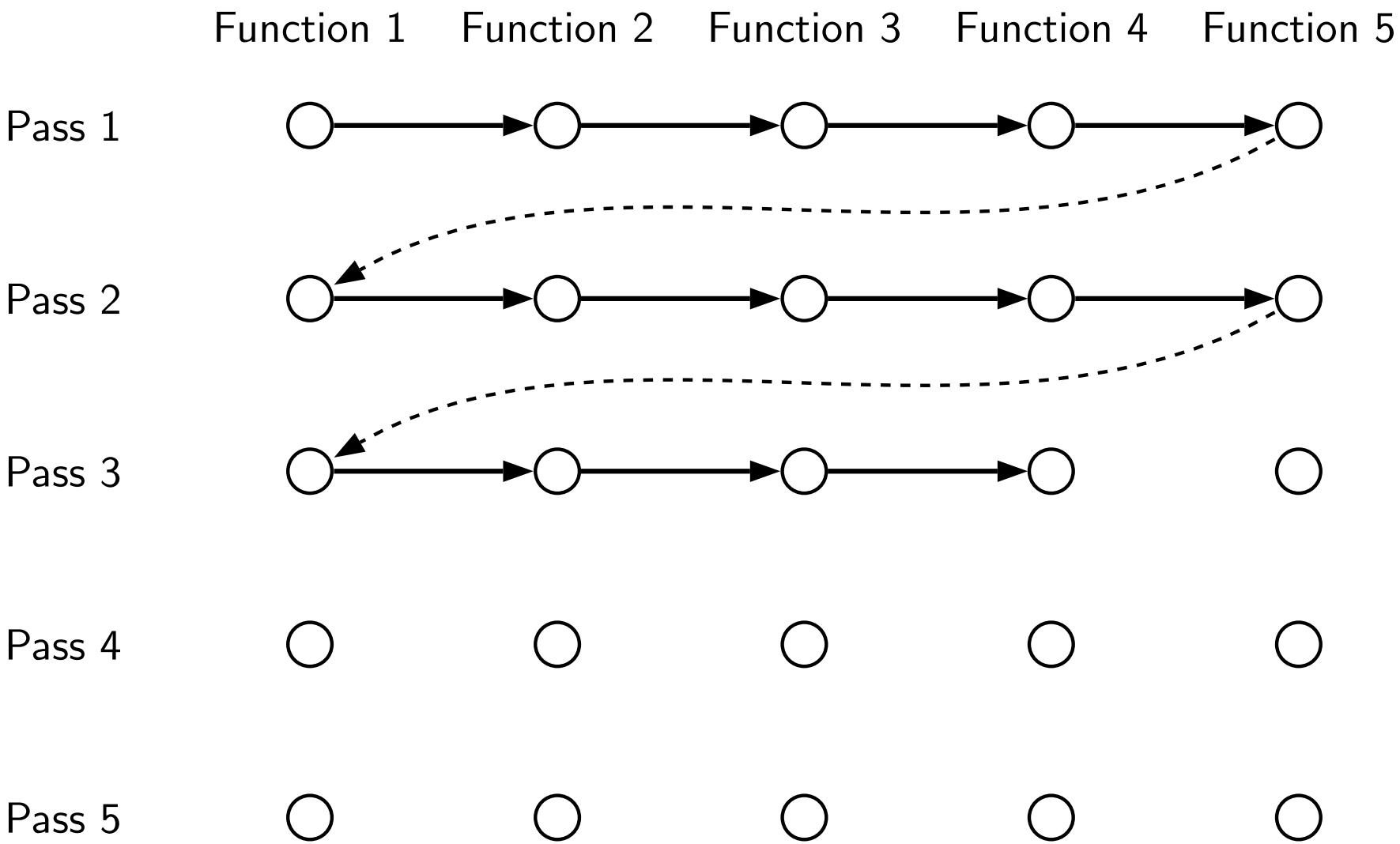
Execution Order in Interprocedural Passes



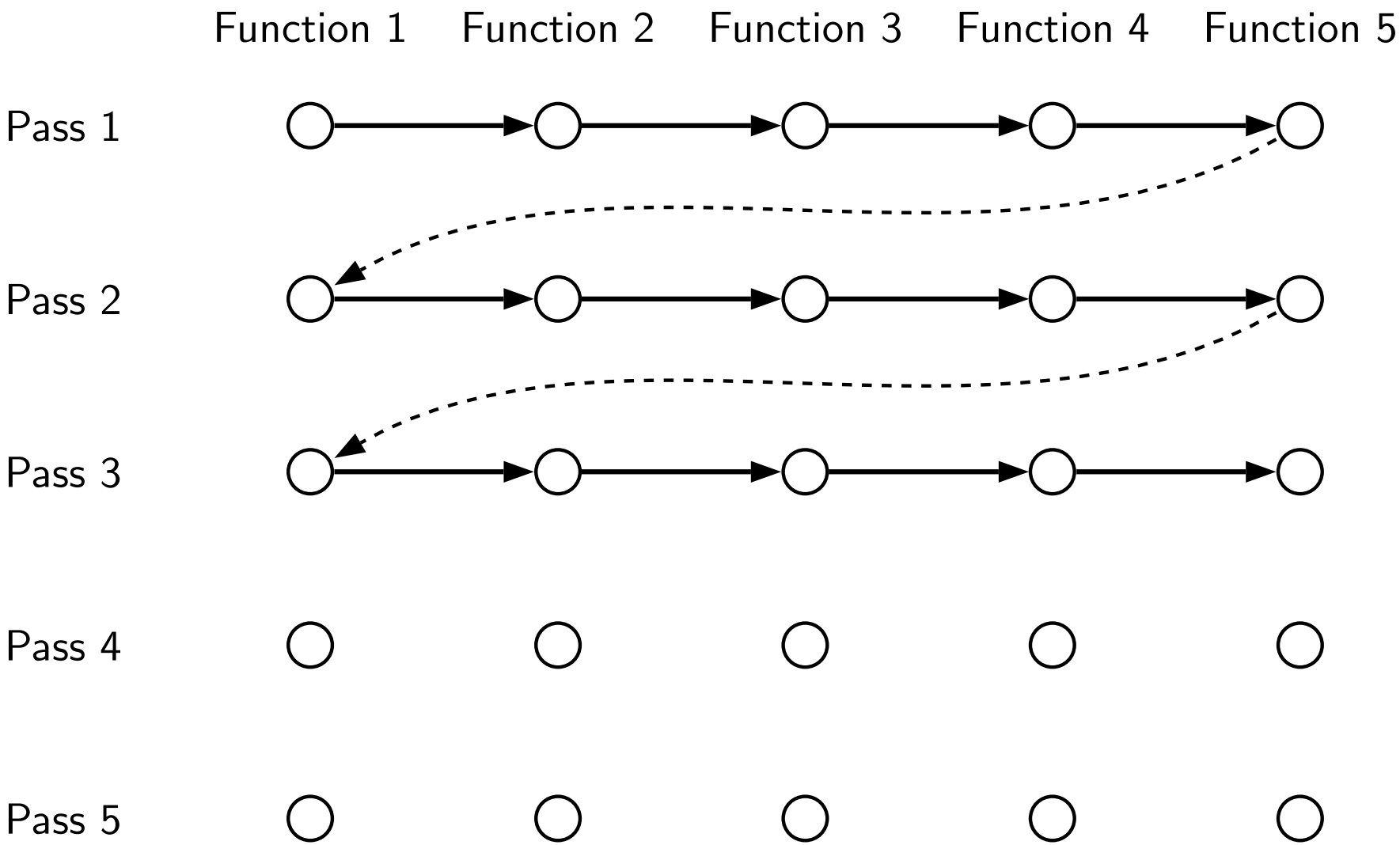
Execution Order in Interprocedural Passes



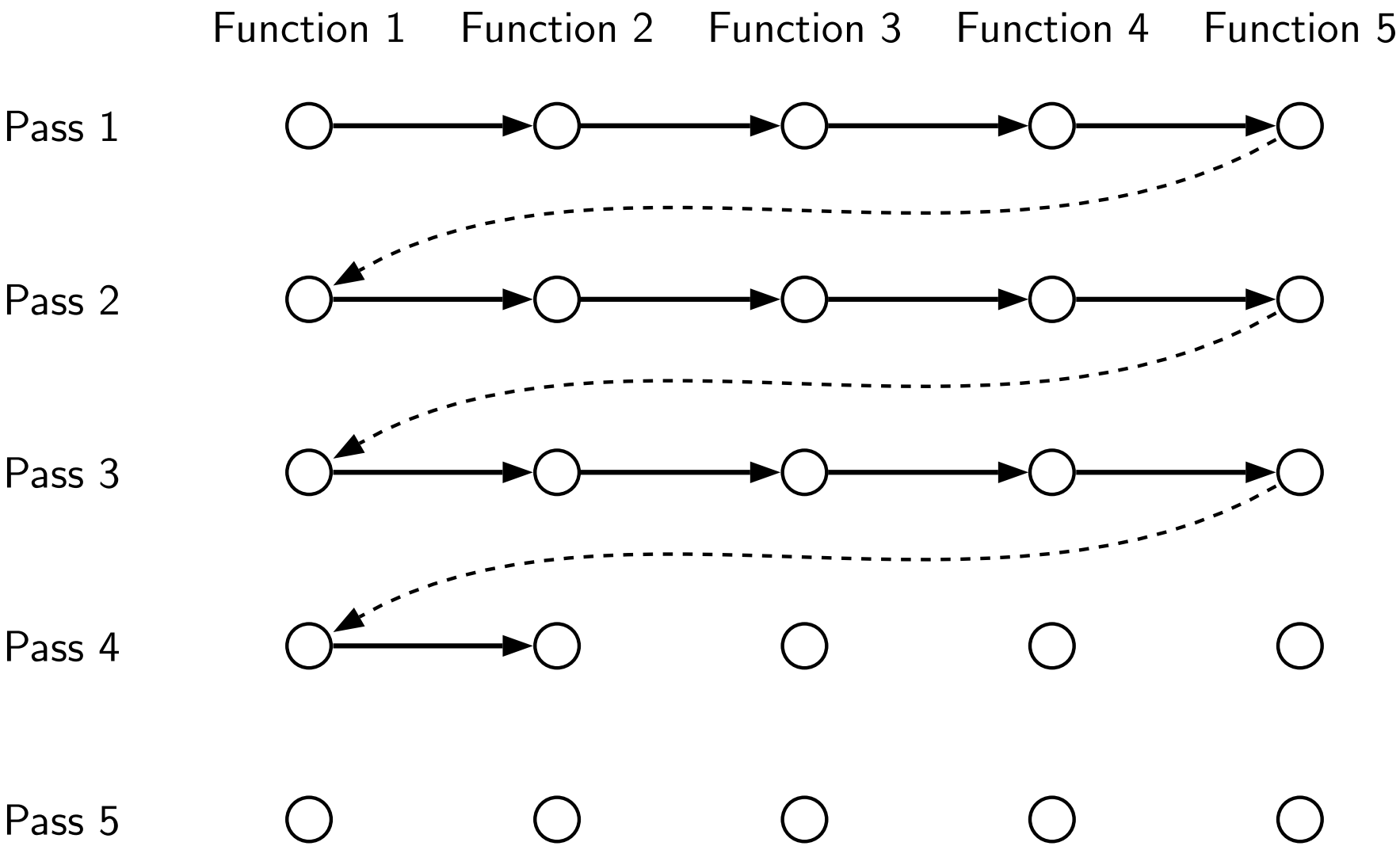
Execution Order in Interprocedural Passes



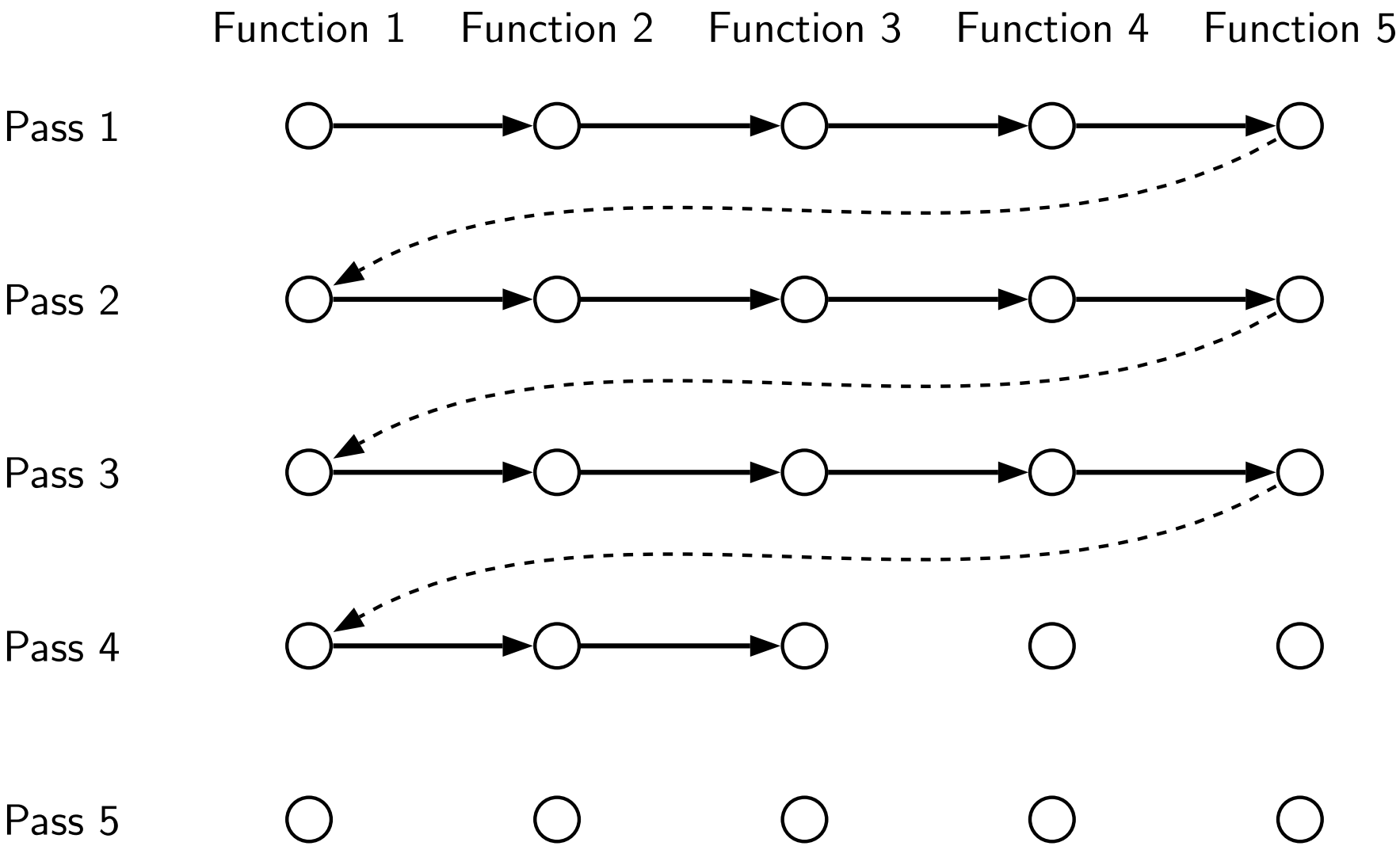
Execution Order in Interprocedural Passes



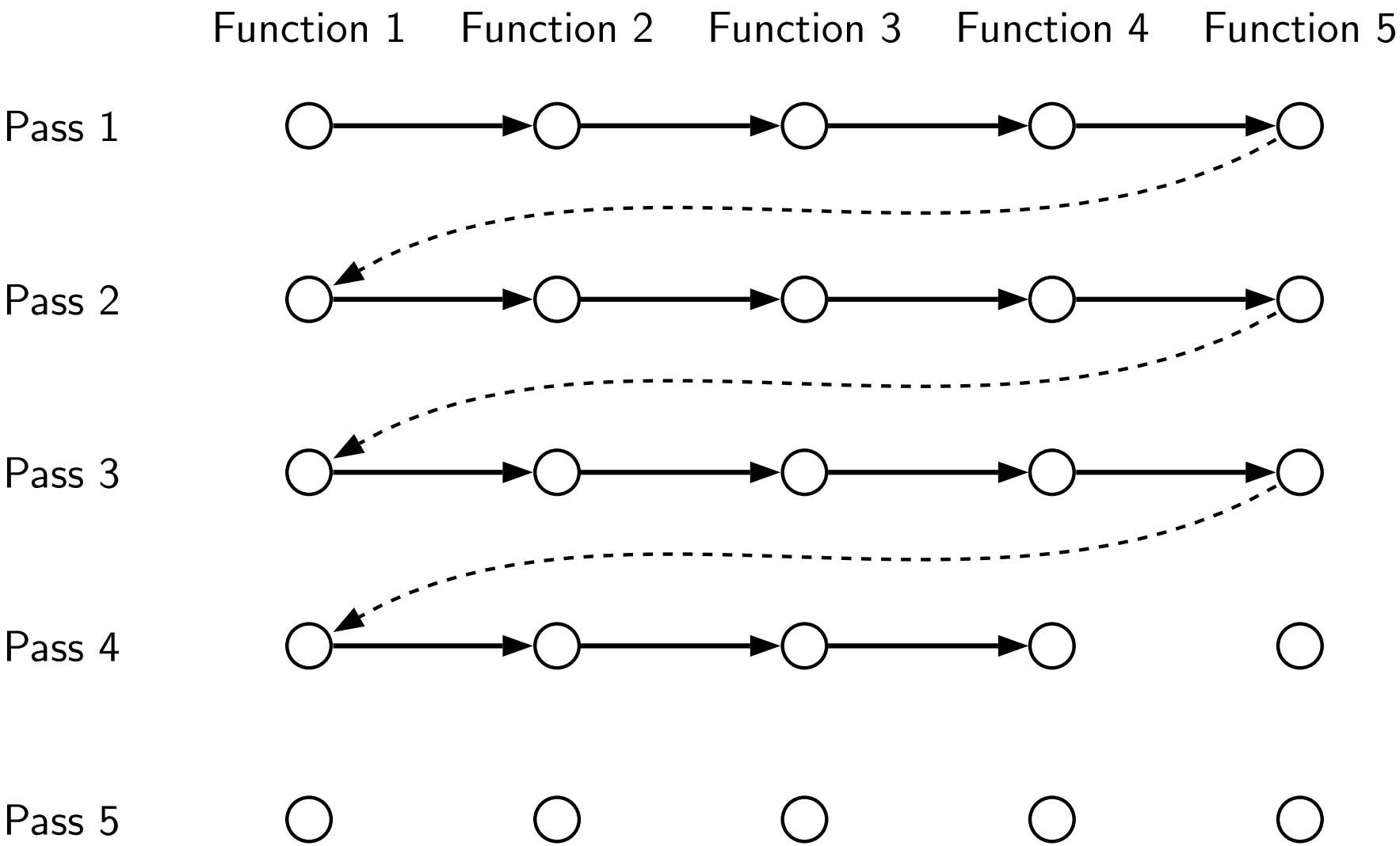
Execution Order in Interprocedural Passes



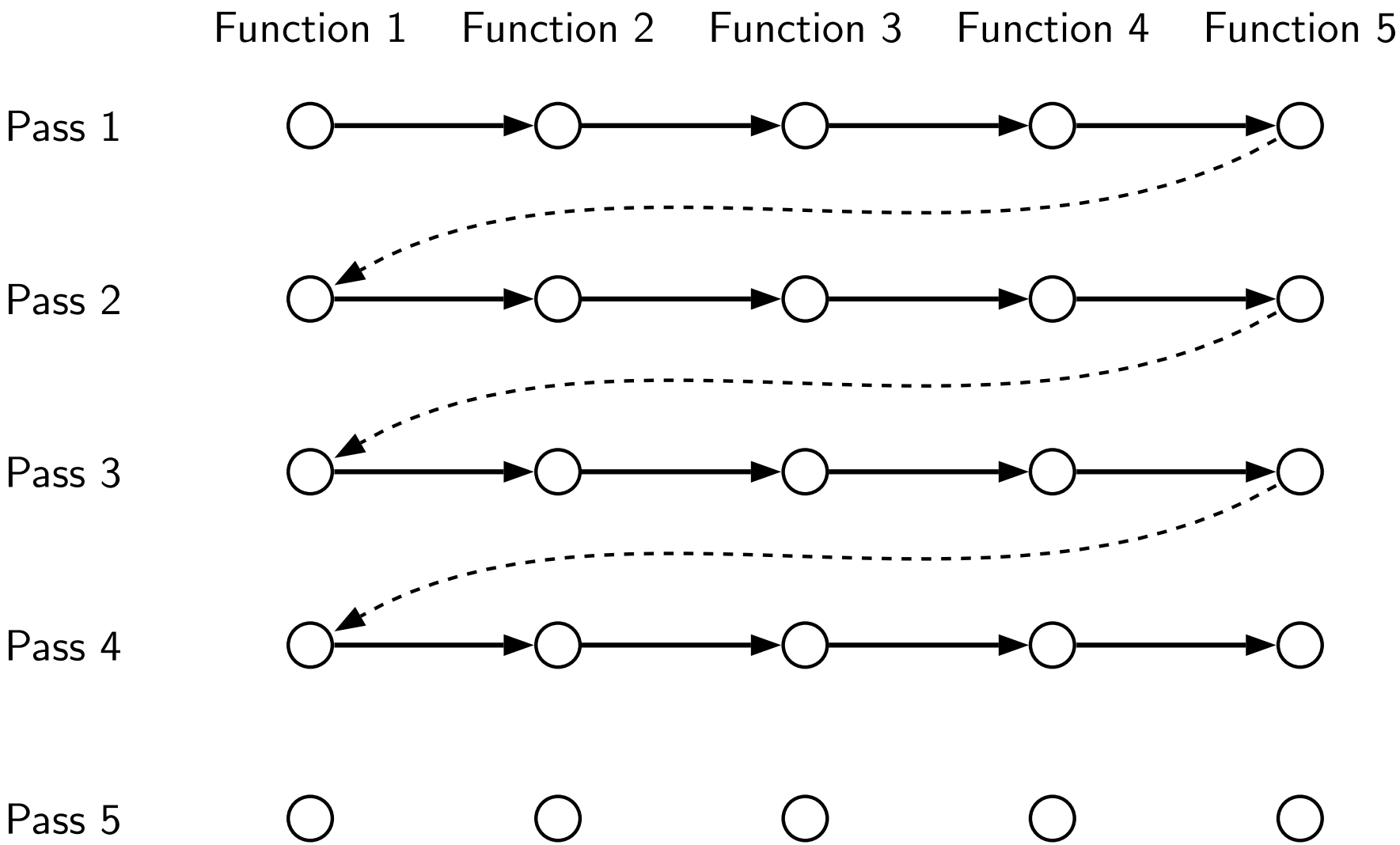
Execution Order in Interprocedural Passes



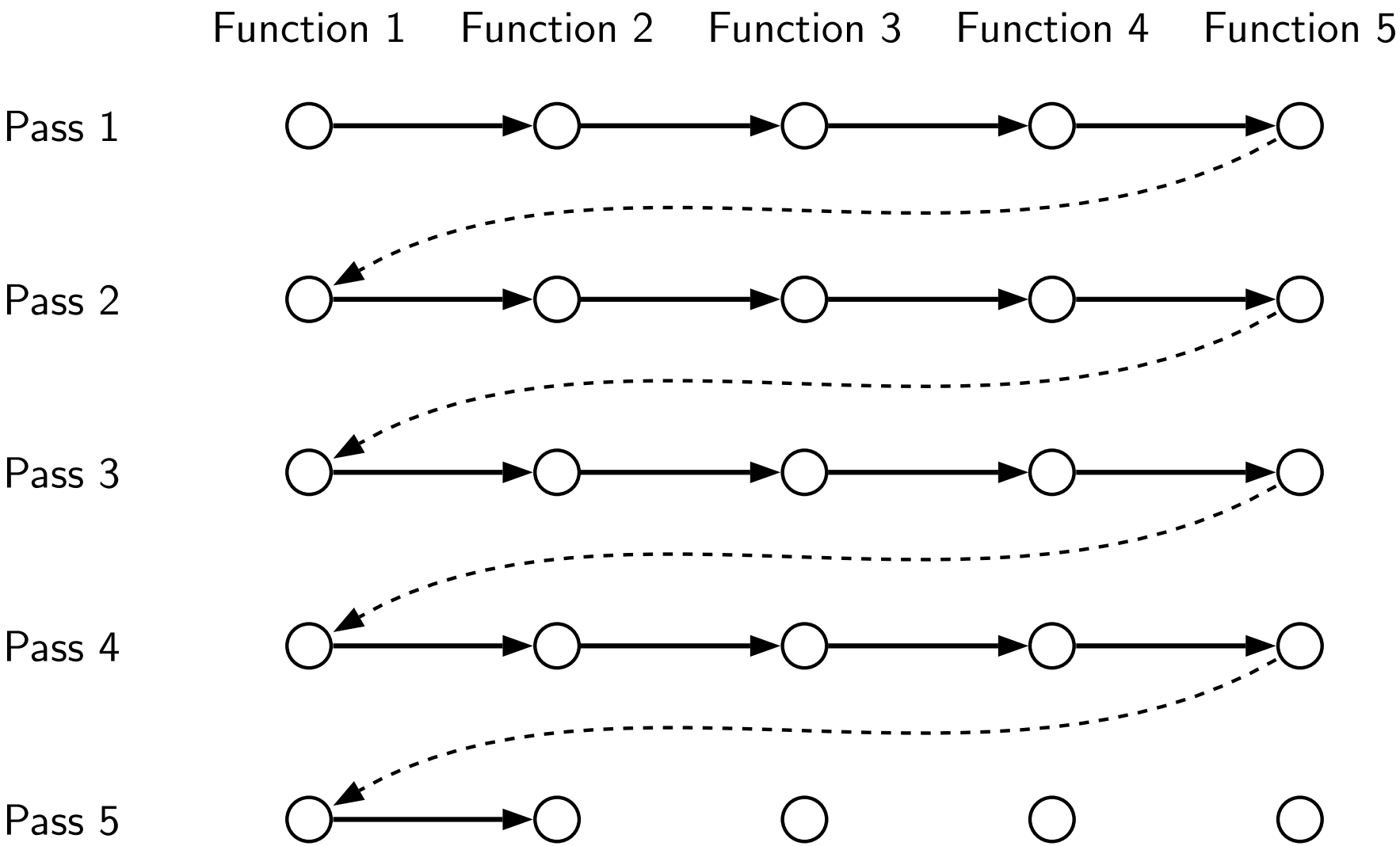
Execution Order in Interprocedural Passes



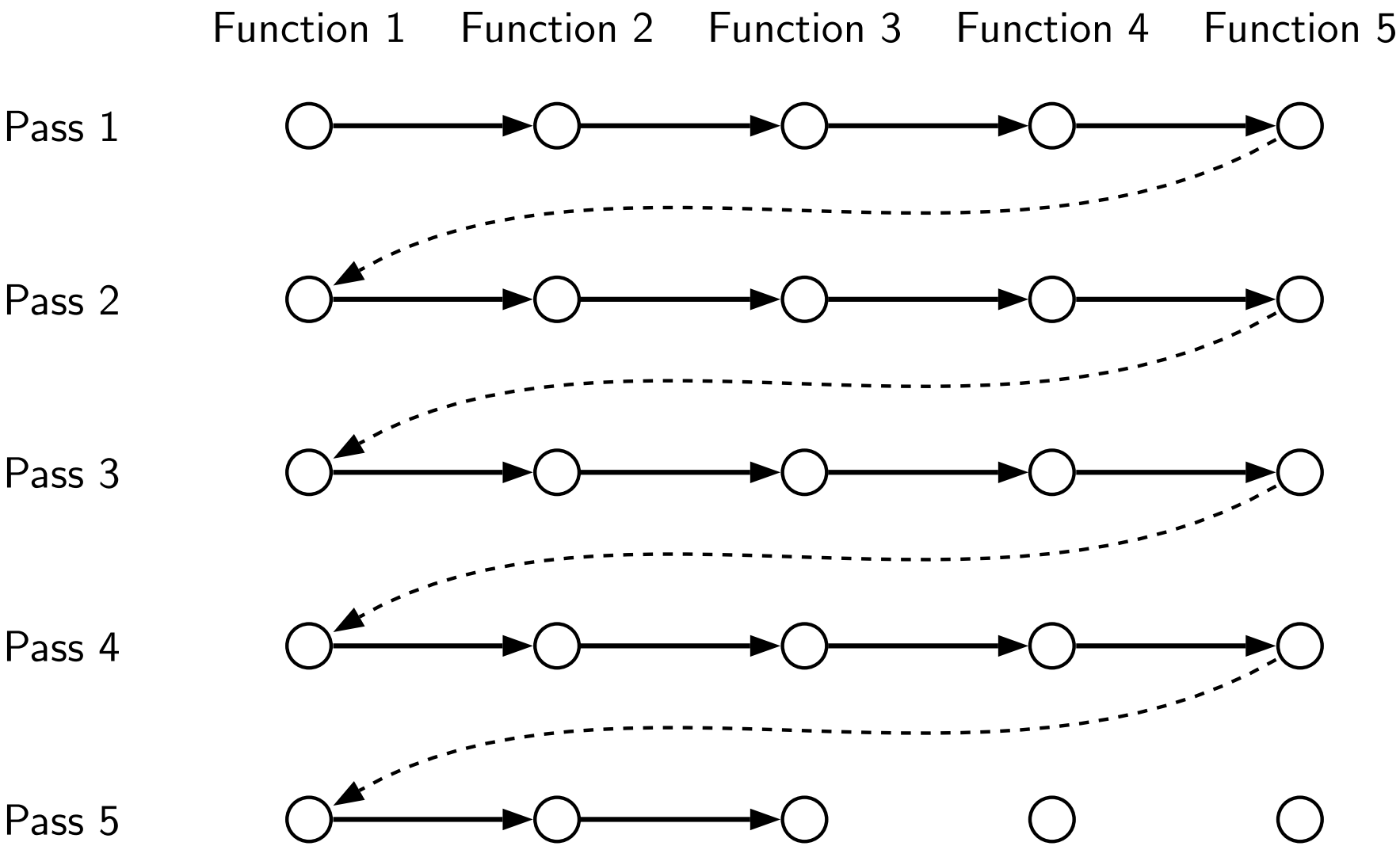
Execution Order in Interprocedural Passes



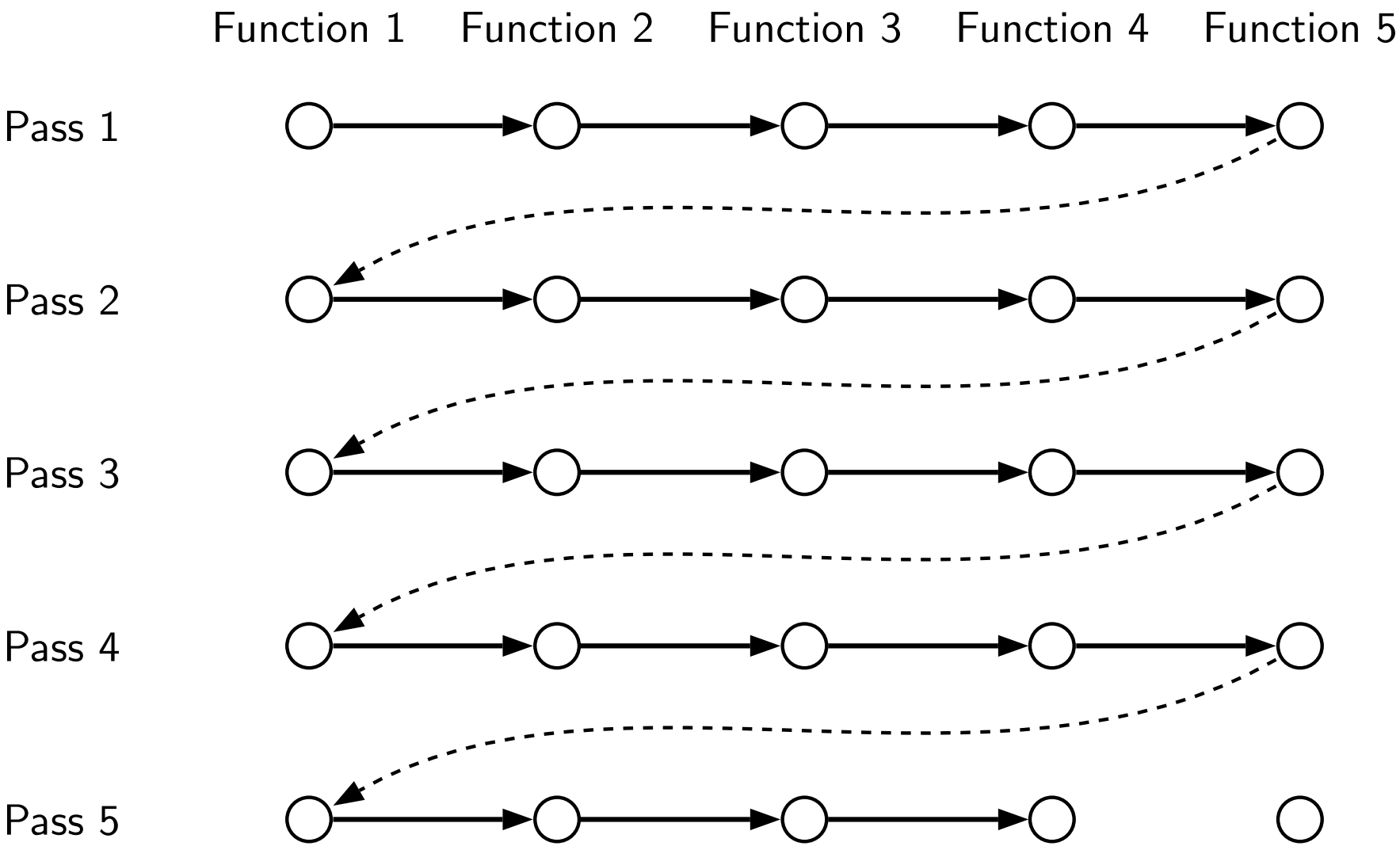
Execution Order in Interprocedural Passes



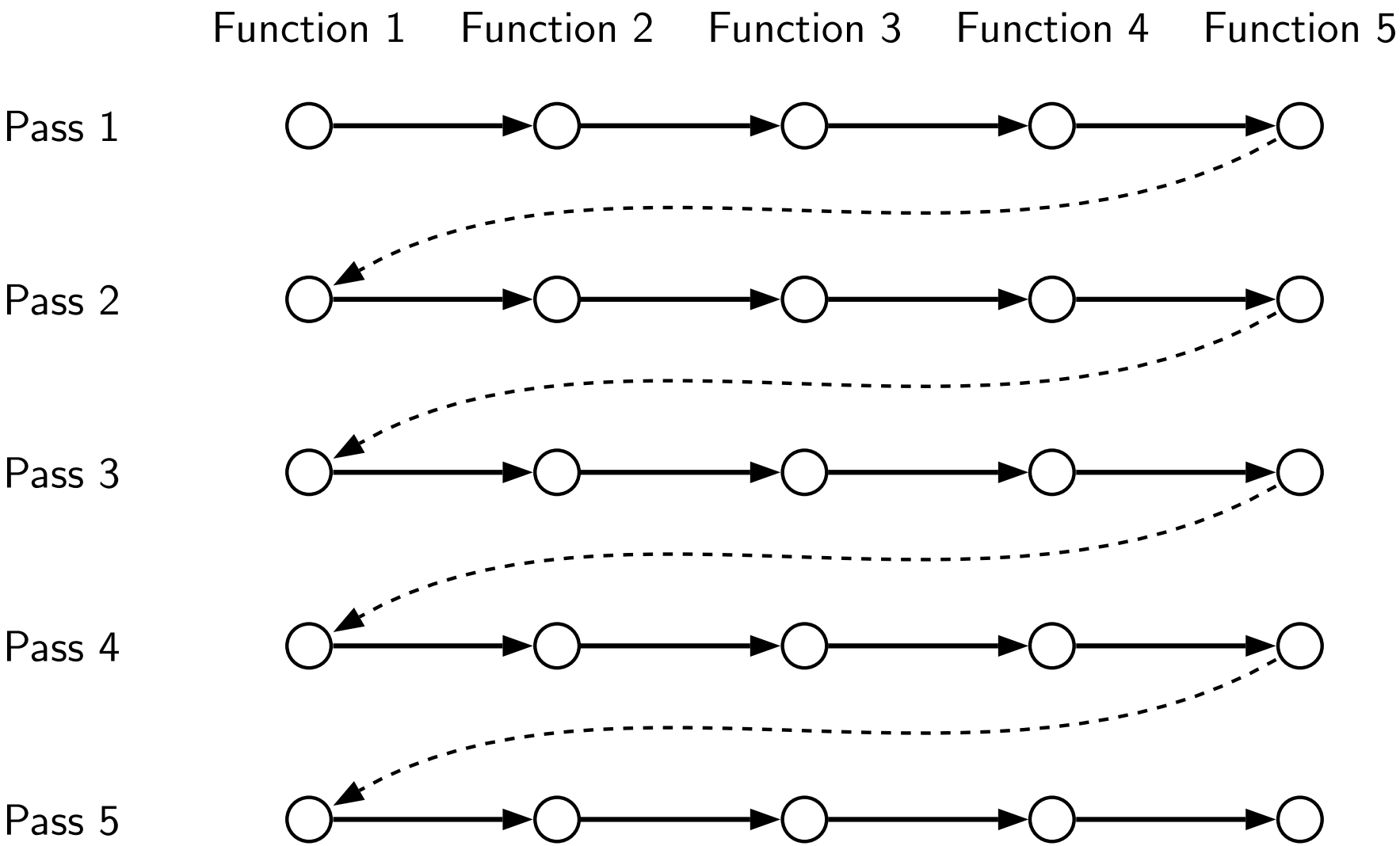
Execution Order in Interprocedural Passes



Execution Order in Interprocedural Passes



Execution Order in Interprocedural Passes



cc1 Control Flow: GIMPLE to RTL Expansion (pass_expand)

```
gimple_expand_cfg
   expand_gimple_basic_block(bb)
      expand_gimple_cond(stmt)
      expand_gimple_stmt(stmt)
         expand_gimple_stmt_1 (stmt)
            expand_expr_real_2
               expand_expr   /* Operands */
                  expand_expr_real
               optab_for_tree_code
               expand_binop /* Now we have rtx for operands */
                  expand_binop_directly
                    /* The plugin for a machine */
                    code=optab_handler(binoptab,mode);
                    GEN_FCN
                    emit_insn
```


cc1 Control Flow: GIMPLE to RTL Expansion (pass_expand)

```
gimple_expand_cfg
    expand_gimple_basic_block(bb)
        expand_gimple_cond(stmt)
        expand_gimple_stmt(stmt)
            expand_gimple_stmt_1 (stmt)
                expand_expr_real_2
                    expand_expr   /* Operands */
                        expand_expr_real
                    optab_for_tree_code
                    expand_binop /* Now we have rtx for operands */
                        expand_binop_directly
                          /* The plugin for a machine */
                          code=optab_handler(binoptab,mode);
                          GEN_FCN
                          emit_insn
```