112062623 資工所 陳冠軒

1. How to run my program

先在 **Makefile** 中將 LLVM_CONFIG 的路徑設定好，以及設定要分析的 C code(ex:test1.c、test2.c …)，然後進入我的 hw2 directories，資料夾結構如下圖 1，在此 directories 下執行 make run 即可 run 我的 pass，執行的結果如下圖 2，執行結果會在 terminal 顯示。

```
ubuntu@ubuntu-MS-7A38:~/advanced_compiler/hw2$ tree
.
├── hw2.cpp
├── Makefile
├── test1.c
├── test2.c
├── test3.c
└── test4.c
```

圖表 1

```
ubuntu@ubuntu-MS-7A38:~/advanced_compiler/hw2$ make run
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/clang -Xclang -disable-O0-optnone -fno-discard-value-names
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/clang -shared -o hw2.so hw2.cpp `/home/ubuntu/advanced_comp
lags` -fPIC -fno-rtti `/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --ldflags`
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/opt -disable-output -load-pass-plugin=./hw2.so -passes=hw2
S1:--------------------
TREF: {}
TGEN: {p}
DEP: {}
TDEF: {(p, S1)}
TEQUIV: {(*p, x)}

S2:--------------------
TREF: {}
TGEN: {pp}
DEP: {}
TDEF: {(p, S1),(pp, S2)}
TEQUIV: {(*p, x),(*pp, p),(**pp, x)}

S3:--------------------
TREF: {pp}
TGEN: {*pp, p}
DEP: {
    pp: S2--->S3
    p: S1-O->S3
}
TDEF: {(p, S3),(pp, S2),(*pp, S3)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}

S4:--------------------
TREF: {p, *pp}
TGEN: {*p, y}
DEP: {
    p: S3--->S4
    *pp: S3--->S4
}
TDEF: {(p, S3),(pp, S2),(*pp, S3),(*p, S4),(y, S4)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}
```

圖表 2

2. Display the output of the released testcases that you can handle

目前可以跑過 4 個 testcase 以下是結果，圖表 3 到圖表 6 分別是資料夾裡 test1.c 到 test4.c 的執行結果，以下是對應表格，都是符合測資的答案。

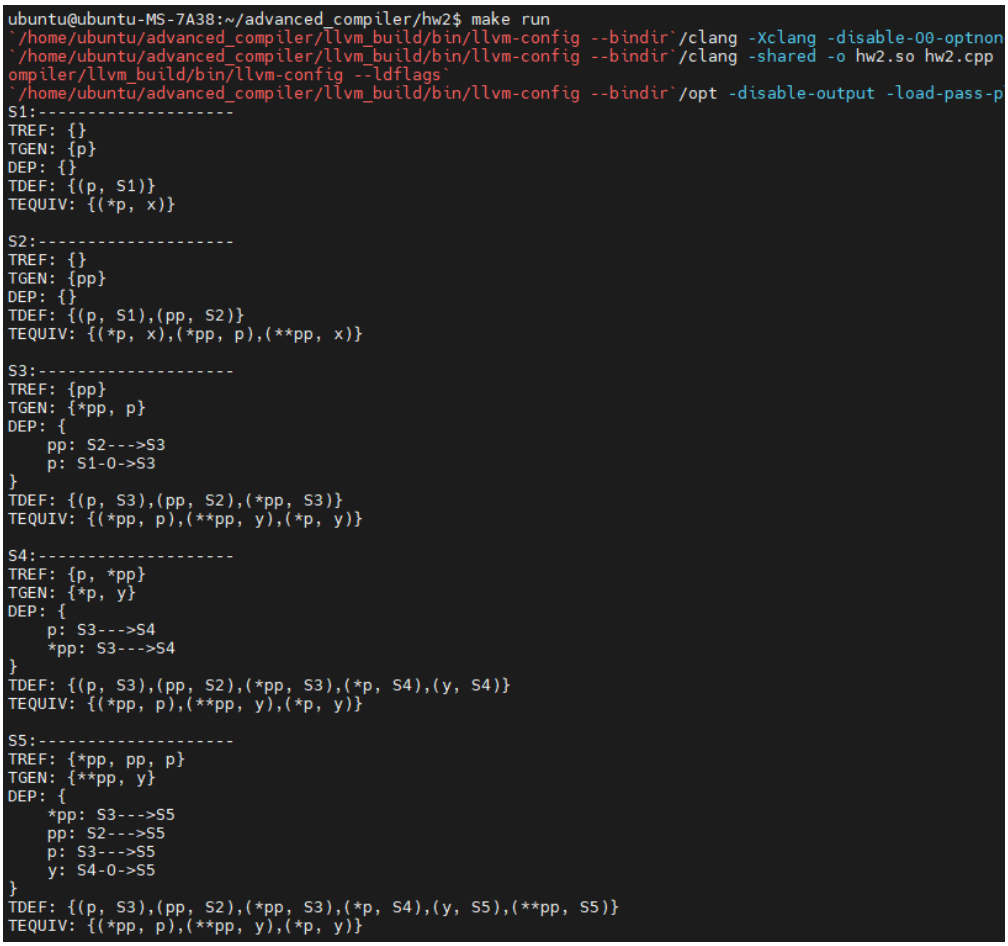
icpp	test1.c	圖表 1
icpp2	test2.c	圖表 2
icpp3	test3.c	圖表 3
foo	test4.c	圖表 4

```
ubuntu@ubuntu-MS-7A38:~/advanced_compiler/hw2$ make run
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/clang -Xclang -disable-O0-optno
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/clang -shared -o hw2.so hw2.cpp
` -fPIC -fno-rtti `/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --ldflags`
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/opt -disable-output -load-pass-
S1:--------------------
TREF: {}
TGEN: {p}
DEP: {}
TDEF: {(p, S1)}
TEQUIV: {(*p, x)}

S2:--------------------
TREF: {}
TGEN: {pp}
DEP: {}
TDEF: {(p, S1),(pp, S2)}
TEQUIV: {(*p, x),(*pp, p),(**pp, x)}

S3:--------------------
TREF: {pp}
TGEN: {*pp, p}
DEP: {
    pp: S2--->S3
    p: S1-0->S3
}
TDEF: {(p, S3),(pp, S2),(*pp, S3)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}

S4:--------------------
TREF: {p, *pp}
TGEN: {*p, y}
DEP: {
    p: S3--->S4
    *pp: S3--->S4
}
TDEF: {(p, S3),(pp, S2),(*pp, S3),(*p, S4),(y, S4)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}
```

圖表 3

```
ubuntu@ubuntu-MS-7A38:~/advanced_compiler/hw2$ make run
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/opt -disable-output -load-pass-plugi
S1:--------------------
TREF: {}
TGEN: {p}
DEP: {}
TDEF: {(p, S1)}
TEQUIV: {(*p, x)}

S2:--------------------
TREF: {}
TGEN: {pp}
DEP: {}
TDEF: {(p, S1),(pp, S2)}
TEQUIV: {(*p, x),(*pp, p),(**pp, x)}

S3:--------------------
TREF: {pp}
TGEN: {*pp, p}
DEP: {
    pp: S2--->S3
    p: S1-0->S3
}
TDEF: {(p, S3),(pp, S2),(*pp, S3)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}

S4:--------------------
TREF: {p, *pp}
TGEN: {*p, y}
DEP: {
    p: S3--->S4
    *pp: S3--->S4
}
TDEF: {(p, S3),(pp, S2),(*pp, S3),(*p, S4),(y, S4)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}

S5:--------------------
TREF: {*pp, pp, *p, p, y, **pp}
TGEN: {**pp, y}
DEP: {
    *pp: S3--->S5
    pp: S2--->S5
    *p: S4--->S5
    p: S3--->S5
    y: S4--->S5
    y: S4-0->S5
}
TDEF: {(p, S3),(pp, S2),(*pp, S3),(*p, S4),(y, S5),(**pp, S5)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}
```

圖表 4


ubuntu@ubuntu-MS-7A38:~/advanced_compiler/hw2$ make run
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/clang -Xclang -disable-O0-optnon
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/clang -shared -o hw2.so hw2.cpp
ompiler/llvm_build/bin/llvm-config --ldflags`
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/opt -disable-output -load-pass-p
S1:--------------------
TREF: {}
TGEN: {p}
DEP: {}
TDEF: {(p, S1)}
TEQUIV: {(*p, x)}
S2:--------------------
TREF: {}
TGEN: {pp}
DEP: {}
TDEF: {(p, S1),(pp, S2)}
TEQUIV: {(*p, x),(*pp, p),(**pp, x)}
S3:--------------------
TREF: {pp}
TGEN: {*pp, p}
DEP: {
pp: S2--->S3
p: S1-O->S3
}
TDEF: {(p, S3),(pp, S2),(*pp, S3)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}
S4:--------------------
TREF: {p, *pp}
TGEN: {*p, y}
DEP: {
p: S3--->S4
*pp: S3--->S4
}
TDEF: {(p, S3),(pp, S2),(*pp, S3),(*p, S4),(y, S4)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}
S5:--------------------
TREF: {*pp, pp, p}
TGEN: {**pp, y}
DEP: {
*pp: S3--->S5
pp: S2--->S5
p: S3--->S5
y: S4-O->S5
}
TDEF: {(p, S3),(pp, S2),(*pp, S3),(*p, S4),(y, S5),(**pp, S5)}
TEQUIV: {(*pp, p),(**pp, y),(*p, y)}

圖表 5

```
ubuntu@ubuntu-MS-7A38:~/advanced_compiler/hw2$ make run
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/clang -Xclang -disable-O0-optnc
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/clang -shared -o hw2.so hw2.cpp
dvanced_compiler/llvm_build/bin/llvm-config --ldflags`
`/home/ubuntu/advanced_compiler/llvm_build/bin/llvm-config --bindir`/opt -disable-output -load-pass-
S1:--------------------
TREF: {b, c}
TGEN: {a}
DEP: {}
TDEF: {(a, S1)}
TEQUIV: {}

S2:--------------------
TREF: {}
TGEN: {p}
DEP: {}
TDEF: {(a, S1),(p, S2)}
TEQUIV: {(*p, y)}

S3:--------------------
TREF: {b, c}
TGEN: {d}
DEP: {}
TDEF: {(a, S1),(p, S2),(d, S3)}
TEQUIV: {(*p, y)}

S4:--------------------
TREF: {a, d, x, y, *p}
TGEN: {f}
DEP: {
    a: S1--->S4
    d: S3--->S4
}
TDEF: {(a, S1),(p, S2),(d, S3),(f, S4)}
TEQUIV: {(*p, y)}

S5:--------------------
TREF: {a, d, x, y, *p}
TGEN: {g}
DEP: {
    a: S1--->S5
    d: S3--->S5
}
TDEF: {(a, S1),(p, S2),(d, S3),(f, S4),(g, S5)}
TEQUIV: {(*p, y)}

S6:--------------------
TREF: {p, i}
TGEN: {*p, y}
DEP: {
    p: S2--->S6
}
TDEF: {(a, S1),(p, S2),(d, S3),(f, S4),(g, S5),(*p, S6),(y, S6)}
TEQUIV: {(*p, y)}
```

```
S6:--------------------
TREF: {p, i}
TGEN: {*p, y}
DEP: {
    p: S2--->S6
}
TDEF: {(a, S1),(p, S2),(d, S3),(f, S4),(g, S5),(*p, S6),(y, S6)}
TEQUIV: {(*p, y)}

S7:--------------------
TREF: {a, d, x, y, *p}
TGEN: {h}
DEP: {
    a: S1--->S7
    d: S3--->S7
    y: S6--->S7
    *p: S6--->S7
}
TDEF: {(a, S1),(p, S2),(d, S3),(f, S4),(g, S5),(*p, S6),(y, S6),(h, S7)}
TEQUIV: {(*p, y)}

S8:--------------------
TREF: {y, *p}
TGEN: {f}
DEP: {
    y: S6--->S8
    *p: S6--->S8
    f: S4-0->S8
}
TDEF: {(a, S1),(p, S2),(d, S3),(f, S8),(g, S5),(*p, S6),(y, S6),(h, S7)}
TEQUIV: {(*p, y)}
```

圖表 6

3. Experiment report

利用 llvm 提供的 api 將 IR 中參數取出，我用到的 api 整理如下

Api name	**用途簡述**
getName()	取得 IR 變數名稱
getOperand()	取得 IR 中的運算元
dyn_cast<Inst_type>()	用於識別指令類型或轉換型別
for (BasicBlock &BB : F)	迭代 basic block
for (Instruction &I : BB)	迭代 IR instruction

先將相關參數取出後，利用 paper 的方法，更新每一輪的 set 就能做出來，以下簡述 paper 中的方法

SET	**定義及更新方法**
TREF	新定義的變數以及 left hand side 的 proper subtree，要新增 TEQUIV 的等價元素進去
TGEN	此 statement 生成出來的定義，left hand side 的 subtree，要新增 TEQUIV 的等價元素進去
DEP	變數相依性
TDEF	當變數在此 statement 有更新時，同時也要更新 reaching definition
TEQUIV	等價的變數集合

較有挑戰性的部分是將參數取出的部分，我的原則是以 STORE 為一個 STATEMENT 的結束，分析前面的 LOAD 指令找出 right hand side 的變數和 left hand side 的變數，找完後根據 paper 中的原則更新 set，這樣就能得到最後結果。