**資工所碩士班一年級 陳冠軒 112062623**

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、test.c)，然後進入我的hw1 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。

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

**圖表 1**

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

**圖表 2**

1. **The case that I can handle**

在我自己的測試中助教給的test1.c跟test2.c都有成功分析出三種dependency，我寫的pass，在迴圈中有A[a\*i+­b]= B[a\*i+b]此種pattern的statement都可以分析，因為我是用陣列資料結構紀錄statement的資訊，目前陣列都是長度為10，所以可以記錄最多10條的statement，程式會自動將每條statement之間的output/flow/anti dependency找出。

1. **Experiment report**

利用llvm提供的api將IR中迴圈的數值、變數的名稱、index的加減乘除參數取出，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 |
| getSExtValue() | 取出IR變數中的整數部分 |
| getOpcodeName() | 得到IR intruction的指令名稱 |

我的方法是一條一條迭代IR指令，迭代過程中取出程式資訊，包含**迴圈起始值**、每個**statement的變數名稱**、**變數index值**，取出後存到陣列中然後利用這些數值分析IR。

迭代instruction過程中如果遇到store的話就當成是一個statement的結束，要開始記錄下一個statement的資訊，如果遇到GetElementPtr的話就是目前IR指令要從source operand切換到操作destination operand，我有維護一個變數(sddiff)專門記錄目前是在操作source或是destination operand，以利我都有確實正確記錄到我想記錄的數值。

還有就是處理運算相關的指令(sub、add、mul)，我用dyn\_cast和getOperand將每條運算相關statemant的參數的整數部分取出，將這些資訊根據目前是在操作source operand還是 destination operand將他們記錄到不同的陣列中，以利後續利用上課教的diophantine公式計算不同statement但相同array變數之間的dependency。

我寫的diophantine function參數中有兩個array中的index(i)的mul數值跟add、sub的值、迴圈的起始值跟結束值、要分析的dependency類型、還有兩個array的變數名稱，有了這些資訊就能用diophantine function計算出是否有dependency以及他們發生的位置，再透過參考剛剛紀錄參數的陣列就能知道是哪兩個statement之間發生dependency，最後用公式計算出發生dependency的位置，一起將這些資訊輸出到terminal。

以下圖3、圖4分別為test1.c和test2.c的實驗結果

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

**圖表 3**

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

**圖表 4**

1. **Bonus**

**Mixing pattern**

我上網查詢相關的CRTP資料發現有以下有點

**第一點是程式碼重用性**，透過繼承模板類別，我們的pass class可以重用共同的功能跟介面，可降低重複的程式碼，並且將pass的結構標準化。

**第二點是可擴展性**，通過繼承PassInfoMixin這樣的class來撰寫pass，可以在現有的模板下製作出更多有特定功能的pass，因此可以說這樣的方法是提升可擴展性。

**第三點是將邏輯實作分離**，在我們的例子中PassInfoMixin這個class提供pass常用的管理邏輯，HW1Pass則專注在實作pass的功能，將邏輯和實作分開可以提供程式碼的可讀性跟維護性。

**最後一點是避免誤用**，將常用的邏輯或是功能包裝起來，需要用時再繼承或是呼叫，這樣的方式可以避免使用者誤用。