ISL Poltting Library with Python API

Hong-Sheng Zheng
Department of Computer Science
Natioal Chiao Tung University
hszheng@cs.nctu.edu.tw

1 Motivation

ISL is a library for manipulating sets and relations of integer ponits bounded by linear constraints. There is already have some related visualizing tools, such like **clint**[8] which is a iteractive visual approach building on the polyhedral model, **islplot**[4] which takes the ISL's data as input, and plots the sets and relations on graph, **pollylabs**[2] which has an online polyhedral playground. However, most existing approaches to plot the isl data has its drawbacks, **clint** can only accept C language as input, aims to provide the user an opportunity to adjust the polytop model in an interactive way, but if you want to do some research on isl, then it may not enough. **islplot** is just what I want, but lots of ISL's data it cannot accept, especially the set of relations. **pollylabs** using D3.js, however, its **islpy**[5] version is a bit old, but you can not change this, because it's just an online playground. So, I want to propose an ISL plotting library using Matplot[1], at least, it should accept more ISL's data than **islplot**, people who are interesting to the polyhedral compilation can benefit.

2 Introduction

ISL is mainly developed by Sven Verdoolage[6], INRIA, is a mathematical software, aims to manipluate sets and relations of integer points bounded by linear constraints, including follow operations: set operations, convex hull, affine hull, integer projection, parametric vertex enumeration, and also include Integer Linear Programming solver.

In recent years, lots of polyhedral compilation tools using ISL as mathematical library, such like LLVM's Polly library[3], Pet: the pollyhedral extracting tool[7], or the iscc which offers an interface to operate the isl structure. The fundermental mathematical ideas behind the polyhedral compilation are Presbuger set, Presbuger map, and Presbuger formula which is a boolean combination of comparisons between Quasi-Affine expressions. In the following secions will describe some of the polyhedral model components: Instance set, Access relations, Dependence relation, and how these components be formulated and plotted, respectively.

2.1 Dynamic execution instances

Pollyhedral compilation will form set of "dynamic execution instances" in the program which boundary described by Presbuger formula, take an example code as shown below, we can label the statement inner the loop as S_1, and the code will form the Presbuger set like:

$$\{S_1(i,j): 1 \leqslant i,j \leqslant 4\}$$

Also called "Iteration domain".

```
for(int i = 1; i < 4; ++i)
    for(int j = 1; j < 4; ++j)
        A[i][j] = A[i-1][j+1];
```

Now the set $\{S_1(i,j): 1 \leqslant i,j \leqslant 4\}$ indicates that what S_1 boundary is, and we need to map a location for this set, we called it "Schedule" which uses Presbuger map, that can describe the above code by:

$$\{S_1(i,j) \to (i,j)\}$$

2.2 Access Relations

The access relations can be described by two set, one is "write set", and the other is "read set", take the code from previous section, the `A[i][j] = A[i-1][j+1]` is writting to `A[i][j]`, and reading from `A[i-1][j+1]`. So the write set and read set can be formulated with following two sets, called "Memory accesses"

$$\begin{aligned} \text{write} &: \{S_1(i,j) \to A(i,j): 1 \leqslant i,j \leqslant 3\} \\ \text{read} &: \{S_1(i,j) \to A(i-1,j+1): 1 \leqslant i,j \leqslant 3\} \end{aligned}$$

2.3 Plotting

The expressions from previous two sections can be ploted like Figure 1. The points in this graph corresponds to each element in $\{S_1(i,j): 1 \leqslant i,j \leqslant 4\}$ by applying its schedule $\{S_1(i,j) \to (i,j)\}$. And the line between two points are called "Dependence", there have three kinds of dependence cause two statments cannot be reordered, read after write(RAW), write after read(WAR), and write after write(WAW), respectively, only the first one is "ture dependence", that means it's cannot be reordered even you apply any schedules on the graph, and the rest dependencies called "false dependence", which may have oppourtinties be eliminated by the polyhedral compiler, and these dependencies can be computed by giving the ISL "Iteration domain", "Schedule", and "Memory accceses".

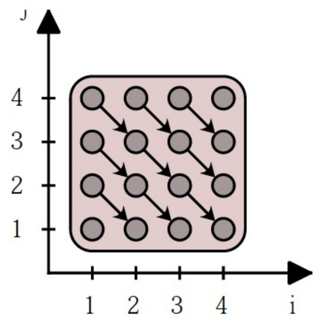

Figure 1. Set of dynamic exectuion instances in program (Credit. Clint)

Intuitively, after writting to $A[i][j]$, this memory location will be passed to the read access at $A[(i+1)-1][(j-1)+1]$ later. let's take the $S_1(1,4)$ as an example, it will have a dependence with $S_1(2,3)$, because the writting at $S_1(1,2)$ is $A(1,2)$ by **write access map**, and this memory location $A(1,4) = A(2-1,3+1)$ will be read from $S_1(2,3)$ by **read access map**.

3 Approaches

The fundermental tool for drawing the graph I choosed is "Matplotlib" which is a Python 2D plotting library, however, because this project I proposed will using the C++ for language selection, so that I choose "matplotlib-cpp" such that I can use the matplot API in the C++ programming enviroment, the basic structure of the program will as shown below Figure 2, the implementation of this project is focus on scislpy whcih is a python API for the scisl, and scisl will call the matplotlib_cpp for helping it to do the plotting thing.

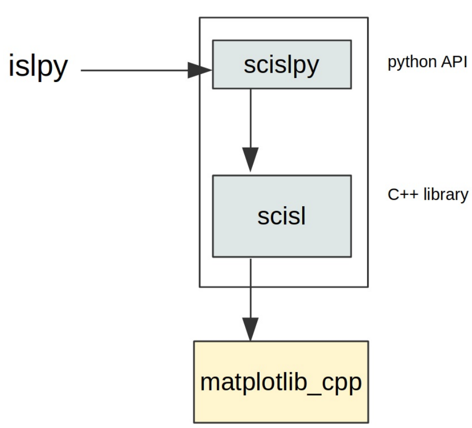

Figure 2. execution flow

4 Timetable

10/28 -12/2	implementation
12/3 - 12/9	writing expermental result
12/10 - 12/30	writing report

REFERENCES

[1] Matplotlib: Python plotting — Matplotlib 3.1.1 documentation.

[2] Polly Labs.

[3] Polly - Polyhedral optimizations for LLVM.

[4] Welcome to islplot's documentation! — islplot 0.1 documentation.

[5] Welcome to islpy's documentation! — islpy 2019.1.2 documentation.

[6] Sven Verdoolaege. Isl: An Integer Set Library for the Polyhedral Model. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Komei Fukuda, Joris van der Hoeven, Michael Joswig, and Nobuki Takayama, editors, *Mathematical Software – ICMS 2010*, volume 6327, pages 299–302. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] Sven Verdoolaege and Tobias Grosser. Polyhedral Extraction Tool. Page 8.

[8] Oleksandr Zinenko, Stephane Huot, and Cedric Bastoul. Clint: A direct manipulation tool for parallelizing compute-intensive program parts. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 109–112. Melbourne, Australia, jul 2014. IEEE.