Python Inspired Machines
PIM

Rajan KhullarID
Tarek Albishara
Mohamad Karaomeroglu

Prof. Wenjia Li

Topics:

Introduction:

C++ programming language is very powerful and is widely used by industry. It provides facilities for low level memory manipulation.

Python is a widely used high level, general purpose, interpreted, dynamic programming language.

Python Inspire Machines (PIM) translate Python-like source code into C++ executable files.

Example:

Python

```python
class box:
    def __init__(self, w, h, l):
        self.width = w
        self.height = h
        self.length = l

    def __str__(self):
        return "this is a box"

    def volume(self):
        return self.width * self.height * self.length


if __name__ == '__main__':
    b = box(1,2,3)
    print b
    print b.volume()
```

C++

```cpp
#include <iostream>
#include <string>

using namespace std;

class box
{
    public:
      box(double w, double h, double l)
        : width(w), height(h), length(l)
        {}

        double width;
        double height;
        double length;

        double volume(){return width * height * length;}
        string output(){return "this is a box";}
};

int main()
{
    box b = box(1, 2, 3);
    cout << b.output() << endl;
    cout << b.volume() << endl;
    return 0;
}
```

BENEFITS OF PIM:

Speed: Developer can write the program on Python which is relatively easy and fast to write and PIM translated into C++.

Security: In Python the developer has to supply the source code, but in C++ the application can be deployed using binary code.

Academia: Student with Python background can implement a code in python and converted to C++.

RELATED WORK:

Cython: converts Python to C.

“Yet Another Compiler Compiler” (YACC) to generate the initial parse tree.

“Toy Parser Generator” (TPG): which is used in this project to parse Python code.

APPROACH:

- Our task is to implement a translator that is able to convert a subset of Python to C++.
- At the start of implementation, we wanted to focus on:
    - Lists
    - Dictionaries
    - Object oriented-ness.

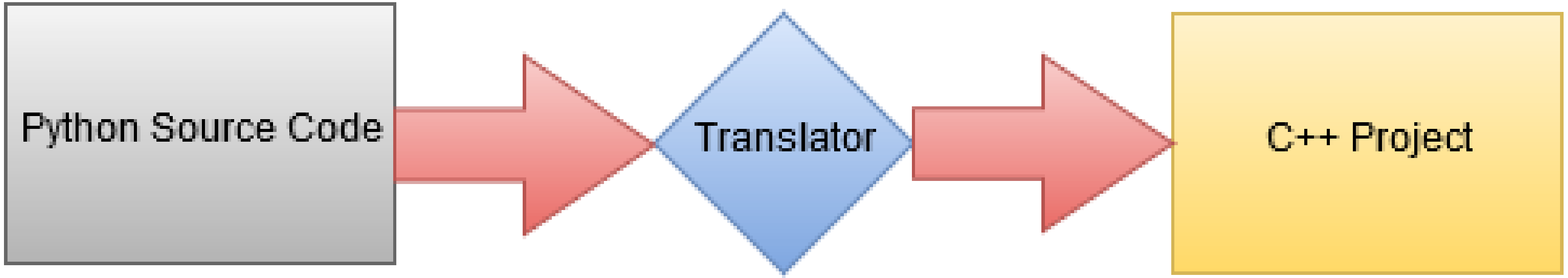
APPROACH:

Initially, we wanted Split up the program into two components:

- Parsing phase: where the program takes Python code and parse it into XML-like format.
- C++ build phase: the program takes XML file and makes it into C++ code.

But we decide <u>NOT TO FOLLOW THIS APPROACH</u>

Initial approach:

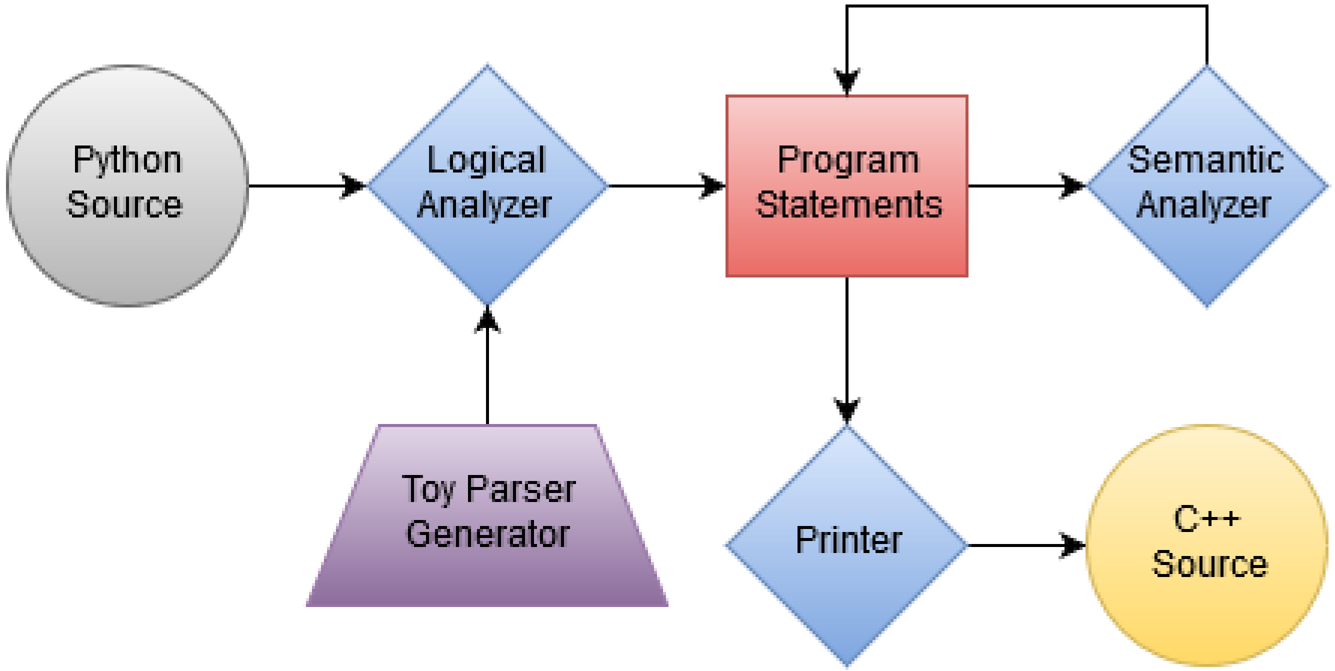


DESIGN:

To implement our Python to C++ translator we had split our tasks into

A. Meta Program
B. Logical Parser
C. Support Modules
D. Top Program

Final Design:



A. Meta Program:

The module describes what programs, variables, and statements are.

The program consists of a list of nodes where each node can be specialized.

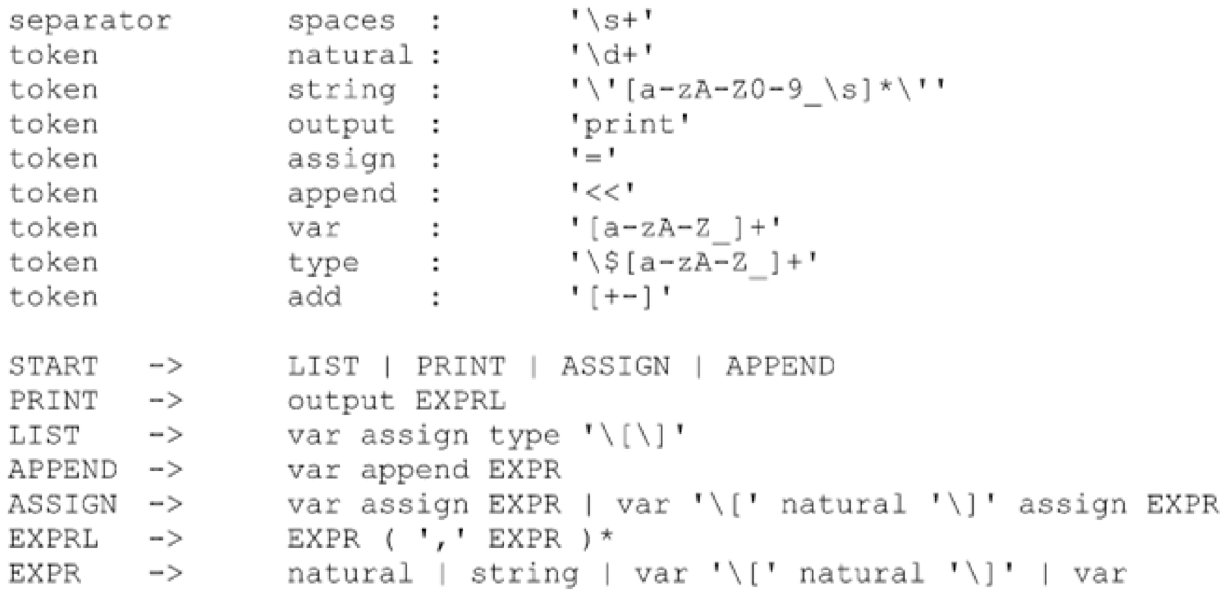
Statement nodes are specialized. In general a statement is an operation string, and a list of arguments.

B. Logical Parser:

Python-like code needs to be parsed into a program object.

Using Toy Parser Generator, we created a module that handles extracting one line of source code into one statement object.

GRAMMAR RULES:



```
separator       spaces  :       '\s+'
token           natural :       '\d+'
token           string  :       '\'[a-zA-Z0-9_\s]*\''
token           output  :       'print'
token           assign  :       '='
token           append  :       '<<'
token           var     :       '[a-zA-Z_]+'
token           type    :       '\$[a-zA-Z_]+'
token           add     :       '[+-]'

START   ->      LIST | PRINT | ASSIGN | APPEND
PRINT   ->      output EXPRL
LIST    ->      var assign type '\[\]'
APPEND  ->      var append EXPR
ASSIGN  ->      var assign EXPR | var '\[' natural '\]' assign EXPR
EXPRL   ->      EXPR ( ',' EXPR )*
EXPR    ->      natural | string | var '\[' natural '\]' | var
```

C. Support Modules

There are two other python files now:

- Mata
- Logic

A bridge module wraps some of the functionality from the Meta Module so the Logical Parser can use it easier.

It reads a source file and prints the corresponding C++ statements line by line.

D. Top Program

Linux bash script.

The program takes two terminal arguments.

- The path of the source Python[ish] code.
- The directory to create the C++ project in.

This entity creates the new user specified directory and copies three files.

- Template
- List header
- Makefile.

FUTURE WORK:

- The translator can be further developed to convert code from Python to C instead of C++.
- Each class in an object oriented language can be broken down into two different structures.
  - One structure would actually hold the class data such as a box’s dimensions.
  - Another structure would point to the class functions such as constructing, printing, and calculating a box’s volume.

Demo

https://youtu.be/te1EnfRlYDE

REFERENCES:

[1] “Welcome to Python.org.” Python.org. Web. 02 Mar. 2016.

[2] “Python Lists.” www.tutorialspoint.com. Web. 23 Mar. 2016.

[3] “Cython: C-Extensions for Python.” Cython: C-Extensions for Python. N.p., n.d. Web. 23 Mar. 2016.

[4] “Cython: The Best of Both Worlds.” IEEE Xplore. N.p.,n.d. Web. 23 Mar. 2016.

[5] “Yacc: Yet Another Compiler-Compiler.” Yacc: Yet Another Compiler-Compiler. N.p., n.d. Web. 06 May 2016