TRANSFORMING PYTHON TO X86_64

COURSE PROJECT : CS335

We would like to thank our professor

Dr. Swarnendu Biswas

& T.A.

Lavanya Sandula

for teaching this course and pushing us to do something which seemed absolutely impossible at first. Bringing this project to fruition was stressful, but fun at the same time. The satisfaction on seeing the code compile correctly was exhilarating.

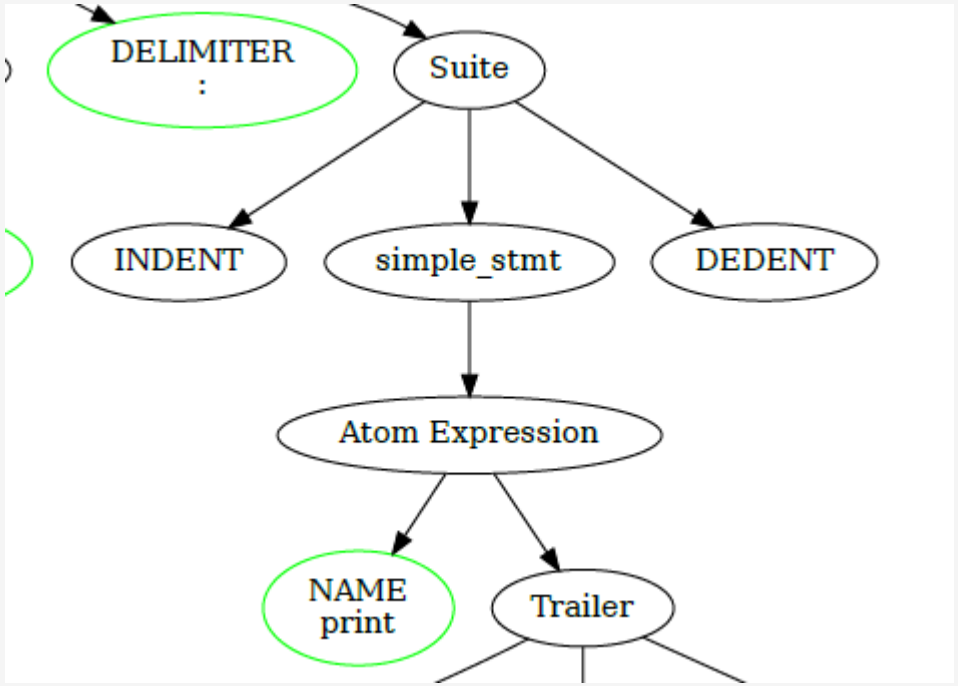
MILESTONE 1

1. Used FLEX for Lexical Analysis
2. Used BISON for Parsing
3. Constructed Parse Tree
4. Constructed AST from the parse Tree by removing nodes with single child
5. Used color coding to improve readability

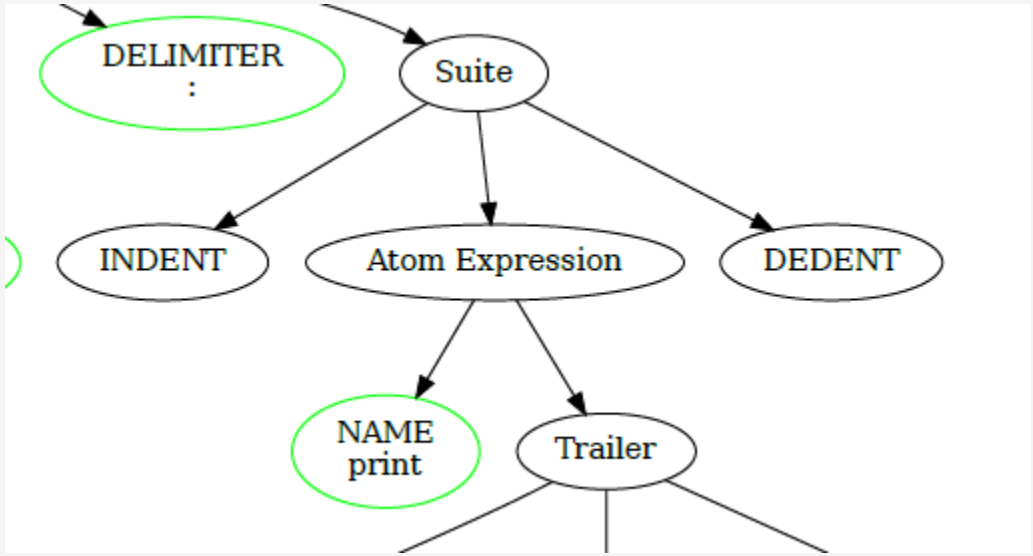
Tree Node Structure

```
struct Node {
    string val;
    string label;
    string category;
    vector<Node*> children;
};
```

```
def test_bracket_argument_list():
    print("All tests passed!")
```



Parse Tree



AST

MILESTONE 2

1. Implemented Symbol Table Data structure
2. Implemented Type checking
3. Generated 3AC code
4. Initiated basic Runtime Support

SYMBOL TABLE STRUCTURE

```
class SYMTAB {
public:
    string tag;
    unordered_map<string, MAPVAL> SYMVAL;
    unordered_map<string,MAPVAL*> freepointers;
    int SYMSCOPE;
    unordered_map<string,SYMTAB*> childs;
    SYMTAB* parent;
```

```
struct MAPVAL {
    string tag;
    int identity;
    int scope;
    int line_no;
    string type;
    string name;
    int size;
    string temp_var;
    // register stuff
    int g_index;
    int reg_name;

    // id
    Value val;

    // array stuff
    vector<Value*> vals;

    // function arguments
    vector<Param*> params;

};
```

MAPVAL STRUCTURE

ACTIVATION RECORD STRUCTURE

```
class ActivationRecord {
public:
    vector<void*> &parameters;
    ActivationRecord* accessLink;
    ActivationRecord* controlLink;
    unordered_map<string, MAPVAL>* localdecs;
    unordered_map<string, MAPVAL*>* freevars;
    unordered_map<string, MAPVAL*> tempo;
    void *returnAddress;
```

```
typedef struct quadruple{
    string op;
    string arg1;
    string arg2;
    string result;
    int index;// target label for this particular instruction
}quadruple;
```

3AC QUADRUPLE STRUCTURE

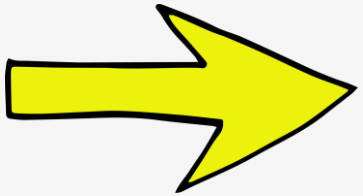
MILESTONE 3

1. MODIFICATIONS IN 3AC
   - Changed printing format
   - Fixed minor bugs
   - Added labels for loops
2. BASIC IMPLEMENTAION TO TRANSFORM 3AC CODE TO X86_64
3. COULDN'T IMPLEMENT FOR LIST CLASSES AND FUNCTION RECURSION

```
def main():
    x:int = 5
    z:int = 10
    add(x,z)
```

```
BeginFunc main
t_5 = 5
x = t_5
t_6 = 10
z = t_6
t_5 = pushparam
x = t_5
t_6 = pushparam
z = t_6
calladd
t_7 = returnpop
EndFunc
```

3AC



```
main:
    pushq %rbp
    movq %rsp, %rbp
    subq $16, %rbp
    movq $5, -16(%rbp)
    movq $10, -24(%rbp)
    movq -16(%rbp), %r8
    pushq %r8
    movq -24(%rbp), %r9
    pushq %r9
    call add
    movq %rax, %r10
    pop %r10
    pop %r8
    pop %r9
    addq $16, %rbp
    movq %rbp, %rsp
    popq %rbp
    ret
```

X86

SUPPORTED FEATURES

- **Variable declarations and assignments** (including support for basic data types like integers, floats, and strings)
- **Basic arithmetic and logical expressions**
- **Control flow statements** (if-else, while, for)
- **Function definitions and calls with recursion support**
- **Compound statements** (e.g., nested if-else, loops)
- **Scoping rules and variable resolution**
- **Type checking and error handling** for type mismatches

UNSUPPORTED FEATURES

DUE TO IMPLEMENTATION COMPLEXITY AND TIME CONSTRAINTS

- Floating-point data type
- Lists
- Classes and object-oriented programming

TEAM

PRASHANT (33.33%)
kprashant21@iitk.ac.in
210750

LAKSHVANT (33.33%)
lakshvant21@iitk.ac.in
210557

HARSH (33.33%)
harshmohan21@iitk.ac.in
210543


THANK
YOU