LFLA

LANGUAGE FOR LINEAR ALGEBRA
TEAM OF GEEKS

• Zhiyuan Guo (Architect)
  • Compiler, Code generation, Semantics

• Guitang Lan (Language Guru)
  • Compiler, Semantic validation, Test case creation

• Jin Liang (Tester)
  • Test case creation, Testing automation, Documentation

• Chenzhe Qian (Manager)
  • Python libraries, Code generation, Documentation
INTRODUCTION

• Is Vector same as Matrix?
• What is Vector Space?
• Why on earth need Matrix?

Why not MATLAB?

How about Python and others?
• Set vector EQUALS matrix
• Mixed math concepts with data structures
GOAL

Math education
Linear algebra programming

Solve $X$ with real math language in computer!

$Ax = B$
TYPES

Featured Primitive Types

- vector
- matrix
- vecspace
- inspace
- affspace

Common Primitive Type

- var
DECLARATIONS

var a = 1.2

vector b = [1,2]

matrix c = [1,2;2,8;]

vecspace d = L([1,2],[3,4])

inspace e = inspace([1,0],[0,1]), c

affspace f = affspace(b, d)
FEATURED OPERATORS

• Belongs @:
  • vector@ vecspace (affspace)

• LieBracket [[ , ]]  
  • [[matrix, matrix]]

• Innerproduct << , >>  
  • id<<vector,vector>>

• Matrix action &  
  • matrix & vector
FEATURED BUILT-IN FUNCTION

- **dim(argument)**
  - argument is vector, vecspace, inspace or affspace

- **basis(vecspace)**
  - return a basis of the vecspace

- **solve(matrix, vector)**
  - return an affspace
  - the solution set of linear equation $m \times x = v$
ARCHITECTURE

Source Code → Scanner → Parser → AST → Type Check

Executable Python File ← Code Gen ← Python AST ← Type Check
TRANSLATION ENVIRONMENT

- **Scope**: symbol_table
- **Functions**: func StringMap.t
- **Global_Variables**: var StringMap.t
- **In_While**: bool
- **In_For**: bool

symbol_table:
- **Parent**: symbol_table
- **Vars**: var StringMap.t
SCOPE RULE

• Global Variable
  • from defined to end

• Function Parameters
  • inside function

• Local variables
  • valid inside the nearest block

```javascript
var i = 4;
function foo(var a)
{
  if a > 2
  {
    var b = a+2;
    print(b);
  }
  else
  {
    var c = 2;
    if a > 0
    {
      c = c+2;
    }
    print(c);
  }
  print(i);
}
function main()
{
  foo(1);
}
```
TYPE CHECK

• Data Type
  • Variable declaration and assign
  • Array declaration and element assign
  • Function parameters pass

• Function return type
  • Our function definition doesn’t declare function return type explicitly, so compiler check return type.

• Control flow
  • Conditional expression type check
```python
#!/usr/bin/python
import sys
sys.path.append('./lib')
from InSpace import *
from AffSpace import *
from Core import *

def othonomalising(bases, ips, n):
    vec=np.array([])
    bases[0] = bases[0] / sqrt(ips.product(bases[0],bases[0]))
    i=0
    for i in range(1, n):
        vec = bases[i]
        j=0
        for j in range(0, i):
            vec = vec - ips.product(bases[i],bases[j]) * bases[j]

        bases[i] = vec / sqrt(ips.product(vec,vec))

    return bases

def main():
    n=2
    v1=np.array([1,2,3])
    v2=np.array([2,3,2,4])
    bases=[v1,v2]
    v4=np.array([1,0,0])
    v5=np.array([0,1,0])
    v6=np.array([0,0,1])
    mat=np.matrix(((1,0,0),(0,1,0),(0,0,1)))
    vecs=[v4,v5,v6]
    ins=InSpace(vecs,mat)
    print(othonomalising(bases, ins, n))

main()
```
TESTING

- Test Suites
- Testing Cases
- Automation Testing
- Test Roles

99 little bugs in the code!
99 little bugs!
You take one down,
You patch it around!
117 little bugs in the code!
AUTOMATION TESTING

• Fast feedback to the team
• Free up time
• A sense of confidence
• An automated script
```python
1 function linearIndep(vector[] vectors, var n)
2 {
3     if n==1 { return 1; }
4     if n > dim(vectors[0])
5         { return 0; }
6
7     vecspace vs;
8     var i;
9     for i = 0:n
10        {
11            if vectors[i]@vs
12                {return 0;}        
13            vs = vs + L(vectors[i]);
14        }
15
16     return 1;
17 }
18
19 function main()
20 {
21     vector v = [1,2,3];
22     vector u = [2,2,3];
23     vector w = v + u;
24     vector x[2] = {v, u};
25     vector y[3] = {v, u, w};
26
27     print(linearIndep(x, 2));
28     print(linearIndep(y, 3));
29 }
```

```python
1 function orthonomalising (vector[] bases, inspace ins, var n)
2 {
3     vector vec;
4
5     bases[0] = bases[0] ./ (sqrt(ips<<bases[0], bases[0]>>));
6
7     var i;
8     for i = 1:n
9        {
10            vec = bases[i];
11            var j;
12            for j= 0:i
13                {
14                    vec = vec - ips<<bases[i], bases[j]>>*.bases[j];
15                }
16            bases[i]=vec./ (sqrt(ips<<vec, vec>>));
17        }
18
19     return bases;
20 }
21
22 function main()
23 {
24     var n = 2;
25     vector v1 = [1,2,3];
26     vector v2 = [2,3, 2, 4];
27
29
30     vector v4 = [1,0,0];
31     vector v5 = [0,1,0];
32     vector v6 = [0,0,1];
33
34     matrix mat = [1,0,0;0,1,0;0,0,1];
36     inspace ins = inspace(vecs, mat);
37
38     print(orthonomalising(bases,ins, n));
39 }
```
HOW WE WORKED

30 Active Issues

28 Closed Issues

2 New Issues
LESSONS LEARNED

• Start EARLY
• Meet regularly
• Plan ahead
• Communication
• Collaboration is key! (Github, Google Drive, WeChat)
Start LIVE DEMO
THANK YOU