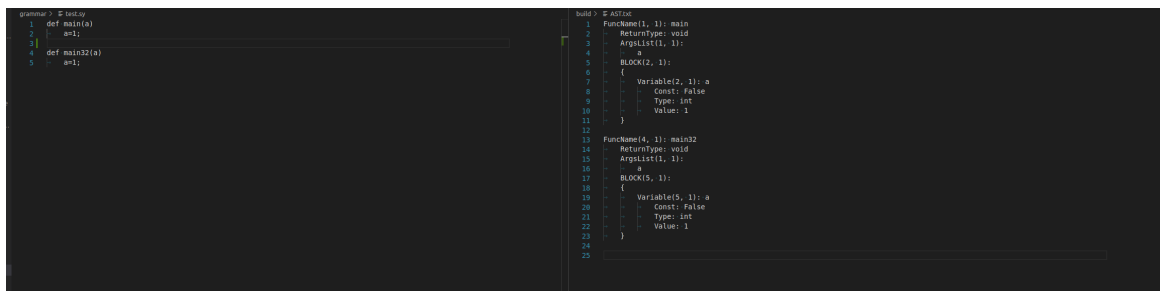


1. 将sy格式的语法规则修改成万花筒的语法规则。

- 支持函数定义
- 支持函数声明
- 支持函数调用
- 支持变量定义， 声明

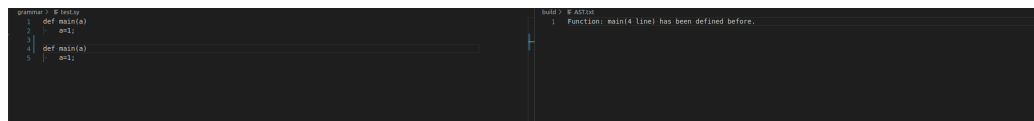


```
grammar: S test;
1 def main(a)
2 | a;
3
4 def main32(a)
5 | a;

build: S Action
1 FuncName(1, 1): main
2 Return type: void
3 ArgList(1, 1):
4 a
5 BLOCK(2, 1):
6 {
7   Variable(2, 1): a
8   Const: False
9   Type: int
10  Value: 1
11 }
12
13 FuncName(4, 1): main32
14 Return type: void
15 ArgList(1, 1):
16 a
17 BLOCK(5, 1):
18 {
19   Variable(5, 1): a
20   Const: False
21   Type: int
22   Value: 1
23 }
24
25
```

2. 万花筒的语义分析

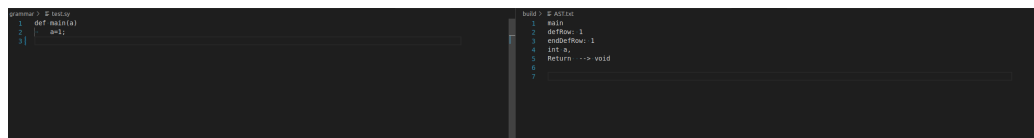
- 重复定义，语义分析时提供报错



```
grammar: S test;
1 def main(a)
2 | a;
3
4 def main(a)
5 | a;

build: S Action
1 Function: main(4 line) has been defined before.
```

- 语法正常时，给出符号表



```
grammar: S test;
1 def main(a)
2 | a;
3
4 def main32(a)
5 | a;

build: S Action
1 main
2 main32
3 endOfMain: 1
4 int a;
5 Return --> void
6
```

3. 利用LLVM 生成中间代码，

还在做，目前了解了LLVM中间代码的规则。