A GRAph JArGon

Zachary Salzbank

Erica Sponsler

Nathaniel Weiss

AGRAJAG's Purpose

- C-based language
- Includes built-in Node data structure
- Ideal for storing graphs/trees
- Data processing

How to use AGRAJAG

- AGRAJAG's syntax is extremely similar to C's
  - Root method instead of main method
  - No easy pointer access
  - No For loops
  - No Strings implemented
    - Functionality is there: nodes of chars.
  - NODES
    - Created with Node<type> = <instance of type>;

What is a Node?

- A node can have a type of either a base type, or another node.
  - Can have Node<Node<Node<…>>> as many times as you would like, as long there is a base type.
- Acceptable base types:
  - Int
  - Char
  - Boolean
- Nodes can have up to 10 children

Examples

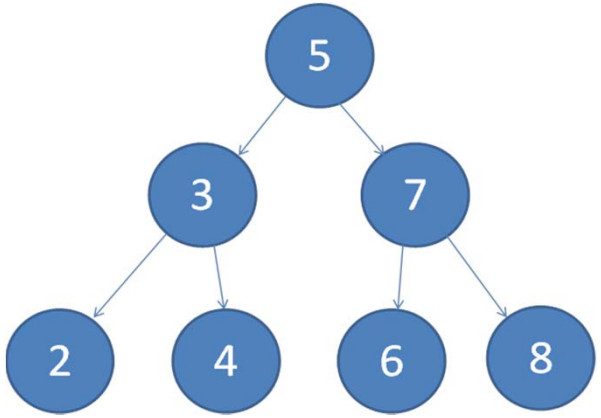
```
1  void root()
2  {
3    Node<Node<int>> x;
4    Node<int> y;
5
6    x = <<42>>;
7    y = x.value;
8    print(x.value.value == y.value);
9  }
```

```
$ ./agrajag < ./tests/test-node-nested-child-value2.ag
true
```

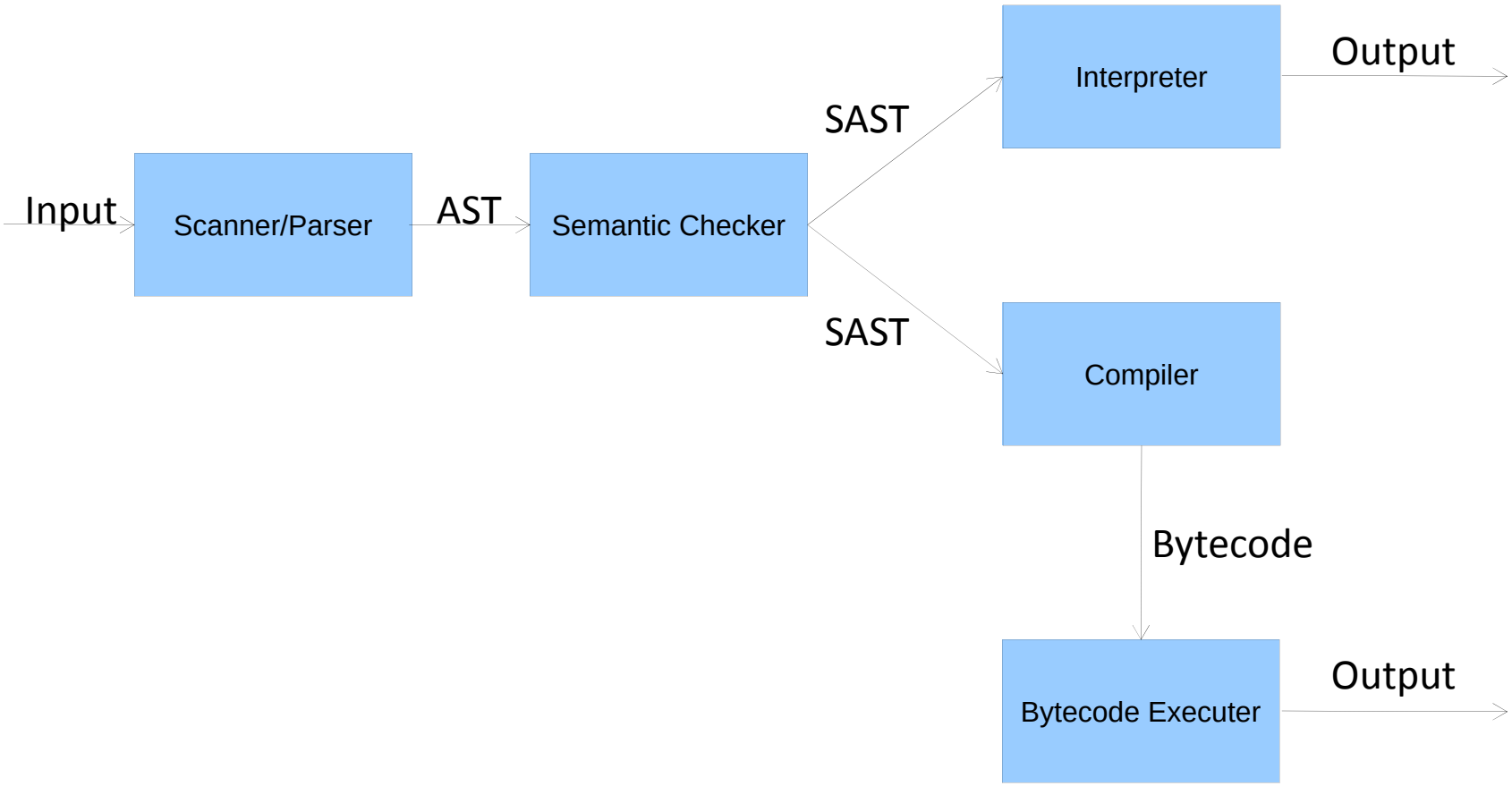
Examples

```
1  void root(){
2      Node<int> treeRoot;
3      Node<int> result;
4
5      treeRoot = <5>;
6      treeRoot[0] = <3>;
7      treeRoot[1] = <7>;
8      treeRoot[0][0] = <2>;
9      treeRoot[0][1] = <4>;
10     treeRoot[1][0] = <6>;
11     treeRoot[1][1] = <8>;
12
13     result = binSearch(treeRoot, 4);
14     if(result == null){
15         print(false);
16     } else {
17         print(true);
18     }
19 }
```

```
22 Node<int> binSearch ( Node<int> sNode, int searchFor ) {
23     while (sNode != null) {
24             if (searchFor < sNode.value) {
25                 sNode = sNode[0];
26             }
27             else {
28                 if(searchFor > sNode.value) {
29                     sNode = sNode[1];
30                 }
31                 else {
32                     return sNode;
33                 }
34             }
35     }
36     return null;
37 }
```



Implementation

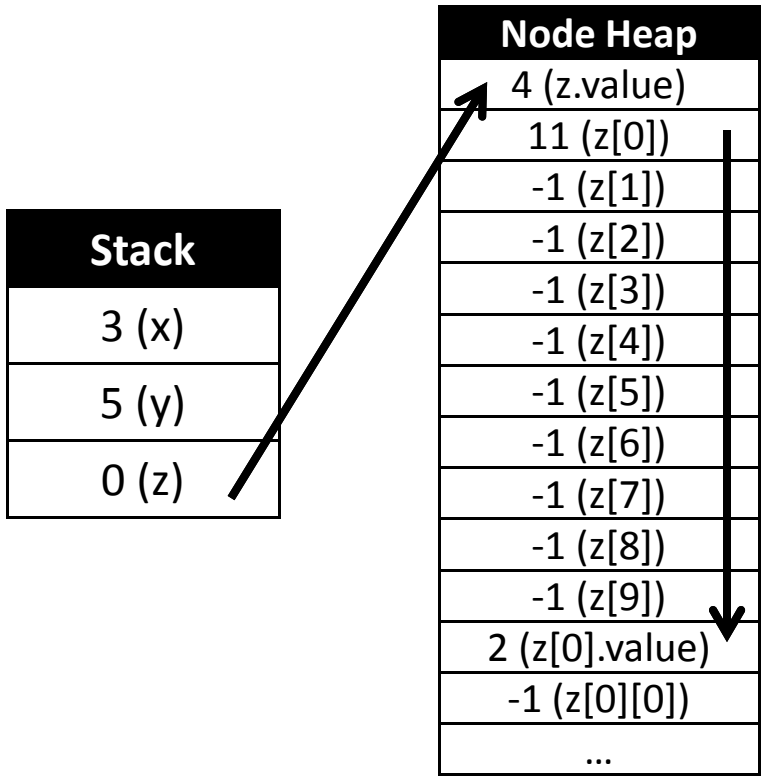


Nodes

- Variables are pointers to objects on ”Node Heap”
- Node Heap stores values and pointers to children

```
int x;
int y;
Node<int> z;

x = 3;
y = 5;
z = <4>;
z[0] = <2>;
```



Bytecode

- Node heap requires new instructions:
  - Ldh – Fetch from heap
  - Sth – Store to heap
  - Cnd – Create node on heap
- Different types differentiate values on the stack between pointers to nodes, and base types.

Lessons Learned

- Teamwork is essential
- Testing implementation details ahead of time
- Project perspective