yeezyGraph

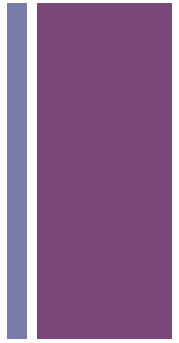
Nancy Xu: System Architect
Wanlin Xie: Project Manager
Yiming Sun: Language Guru

introduction

motivation

- Simple, concise domain-specific language for graph data analysis
  - Syntactic simplicity
  - Structural simplicity
- Out-of-the-box framework
  - Intuitive
- Standardized management of graph data under the hood

language features

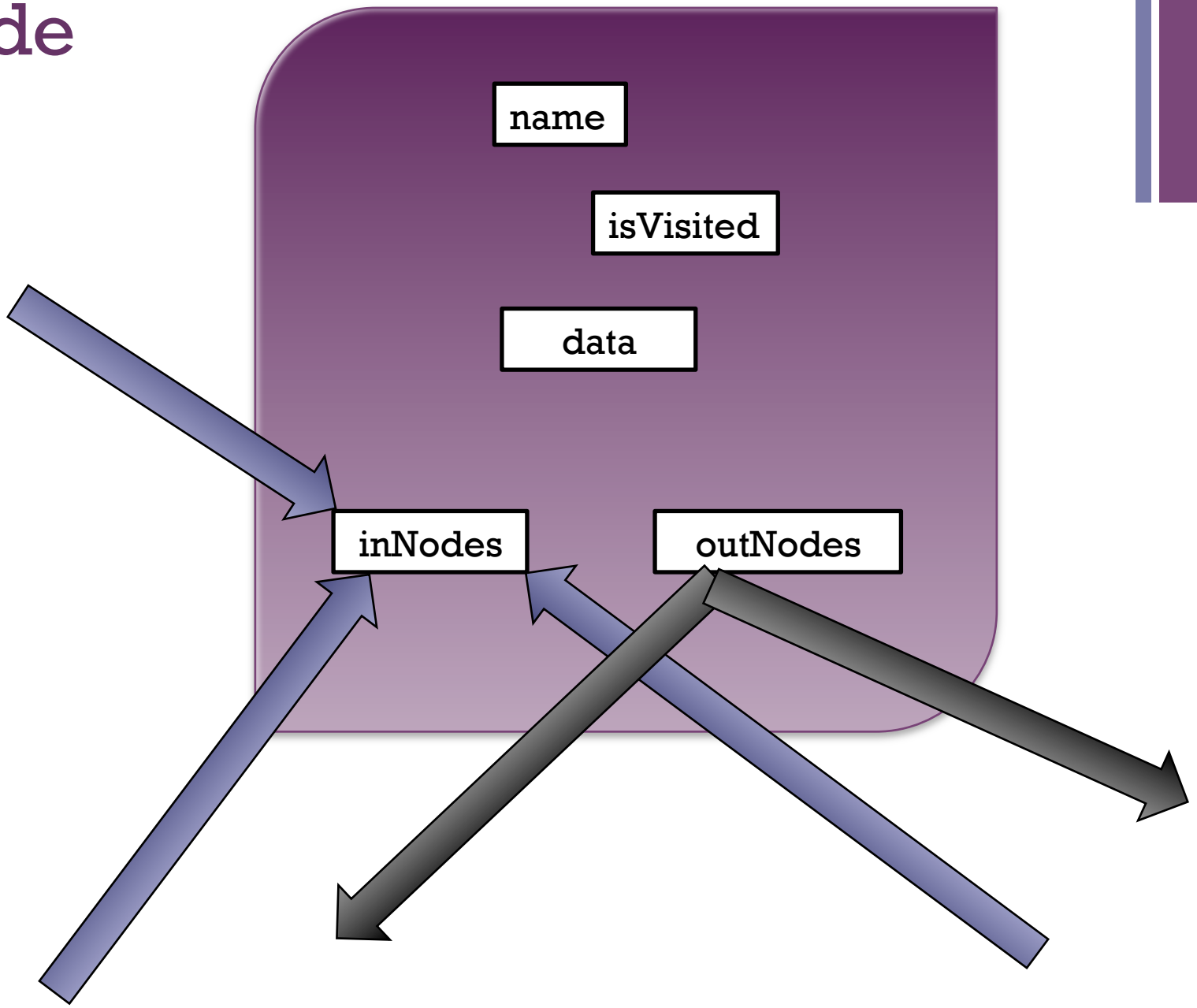


- Node and Graph data types
- Useful standard library
  - Generic list, queue, pqueue data types
- User-defined data types
- Useful error messages
- Compilation to LLVM IR code

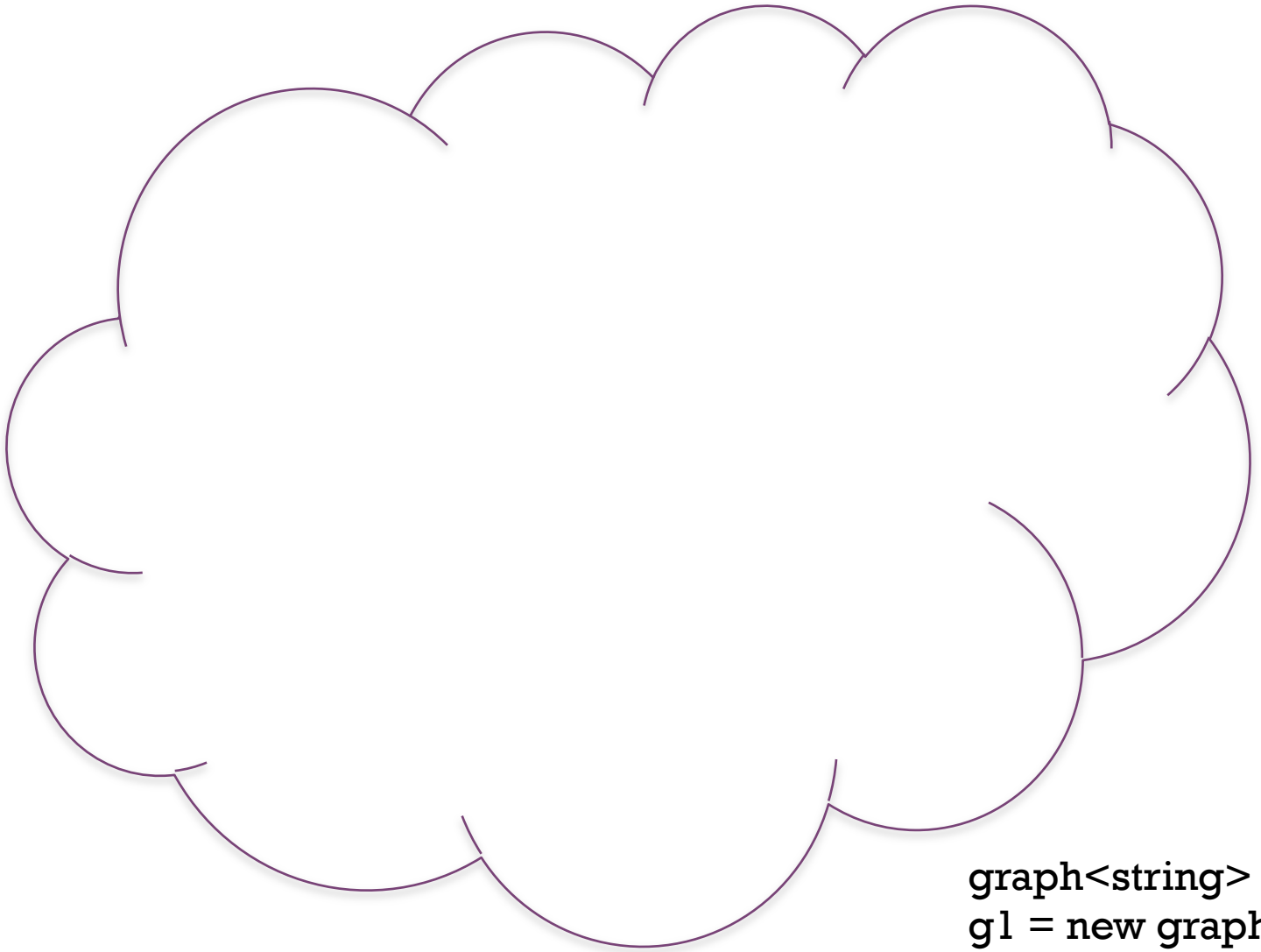
nodes and graphs

node

node

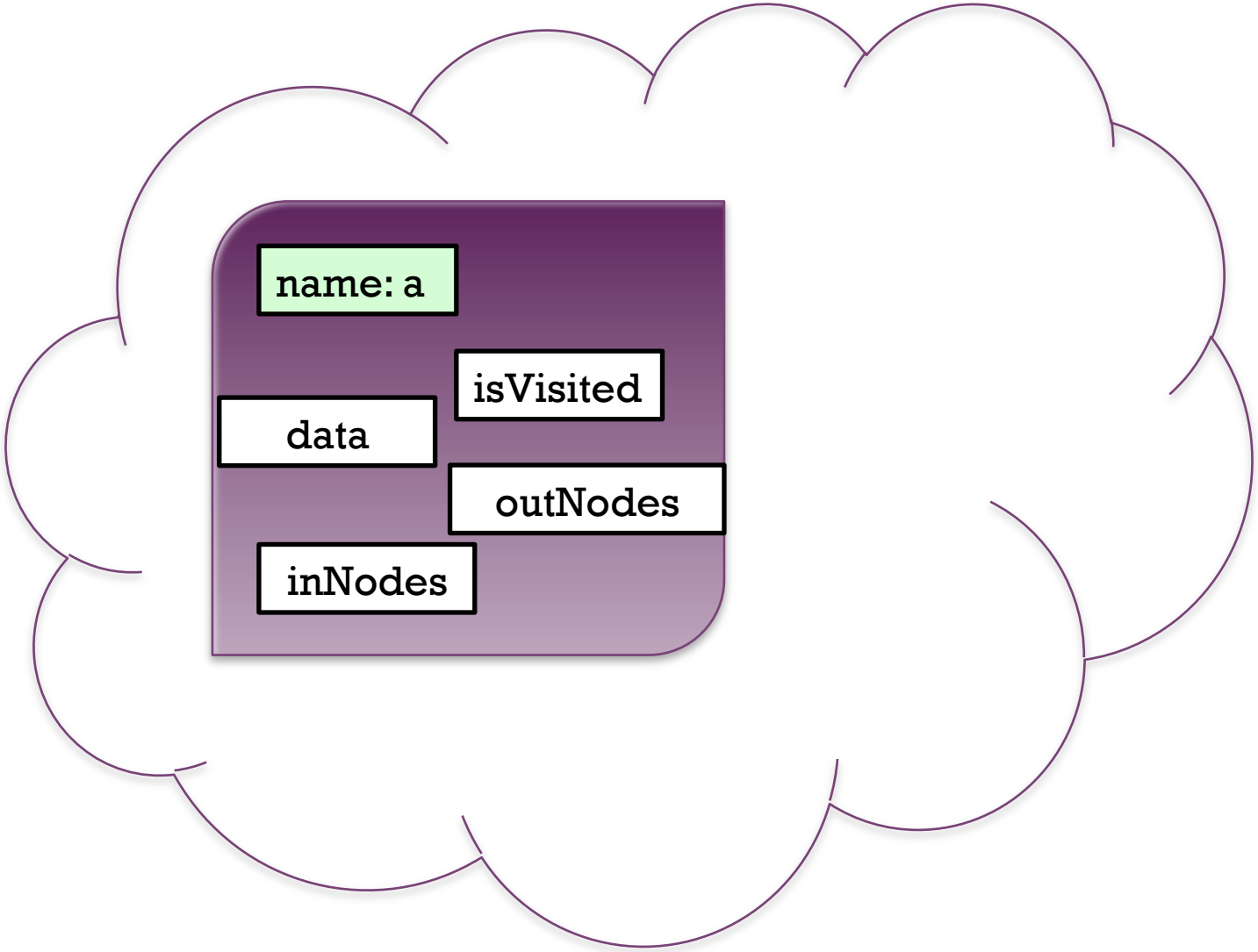


graph



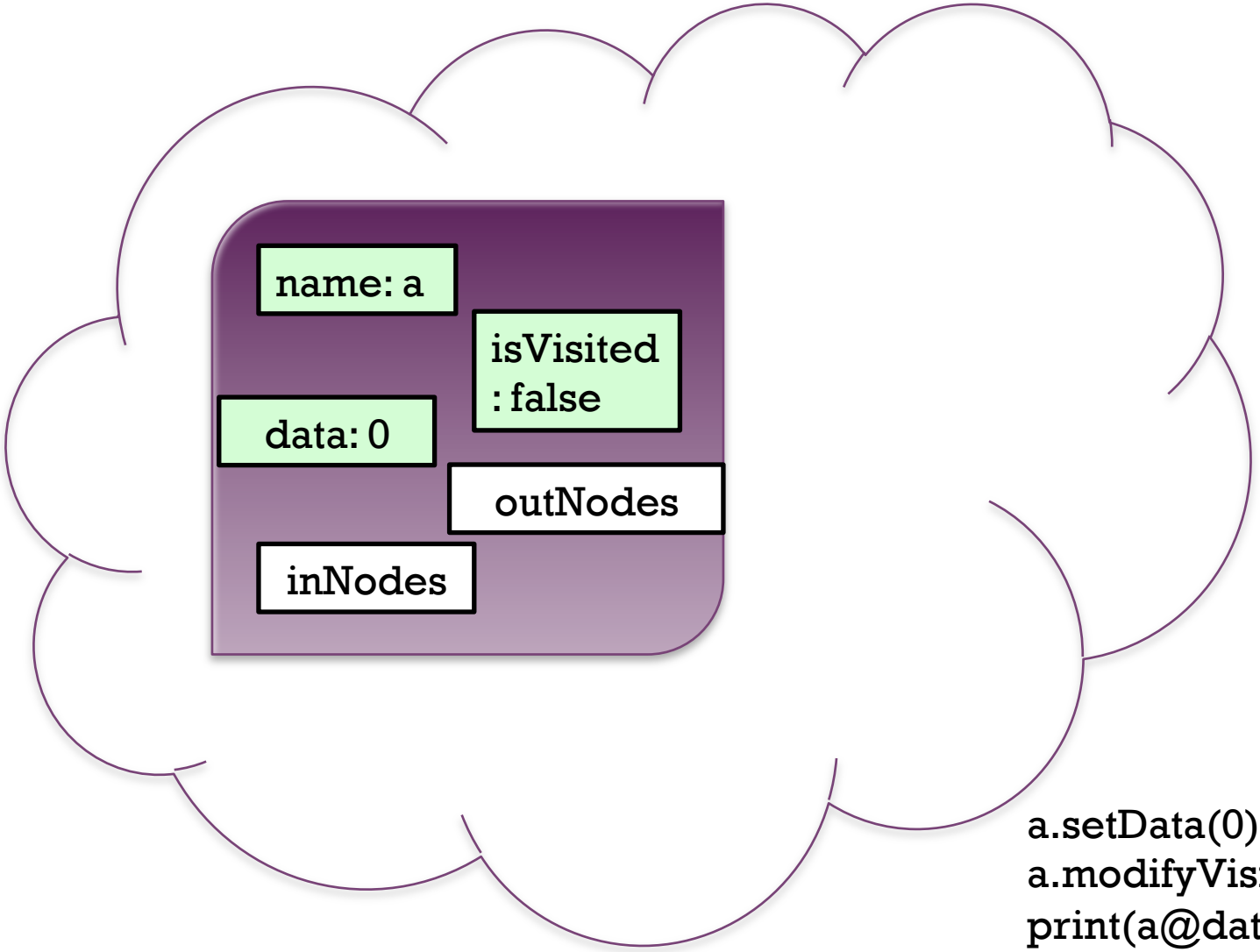
```
graph<string> g1;
g1 = new graph<string>();
```

graph



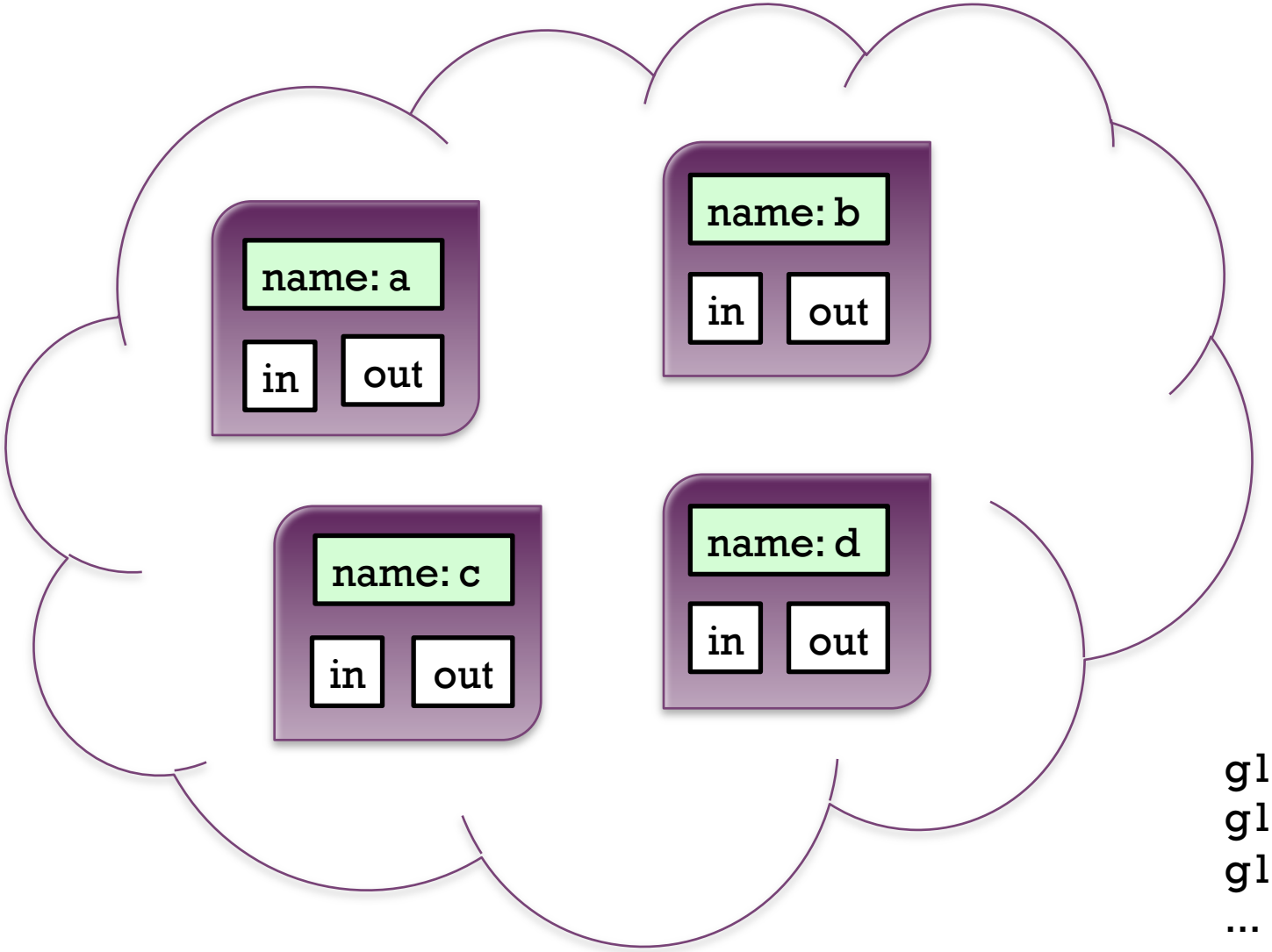
gl~+"a";

graph



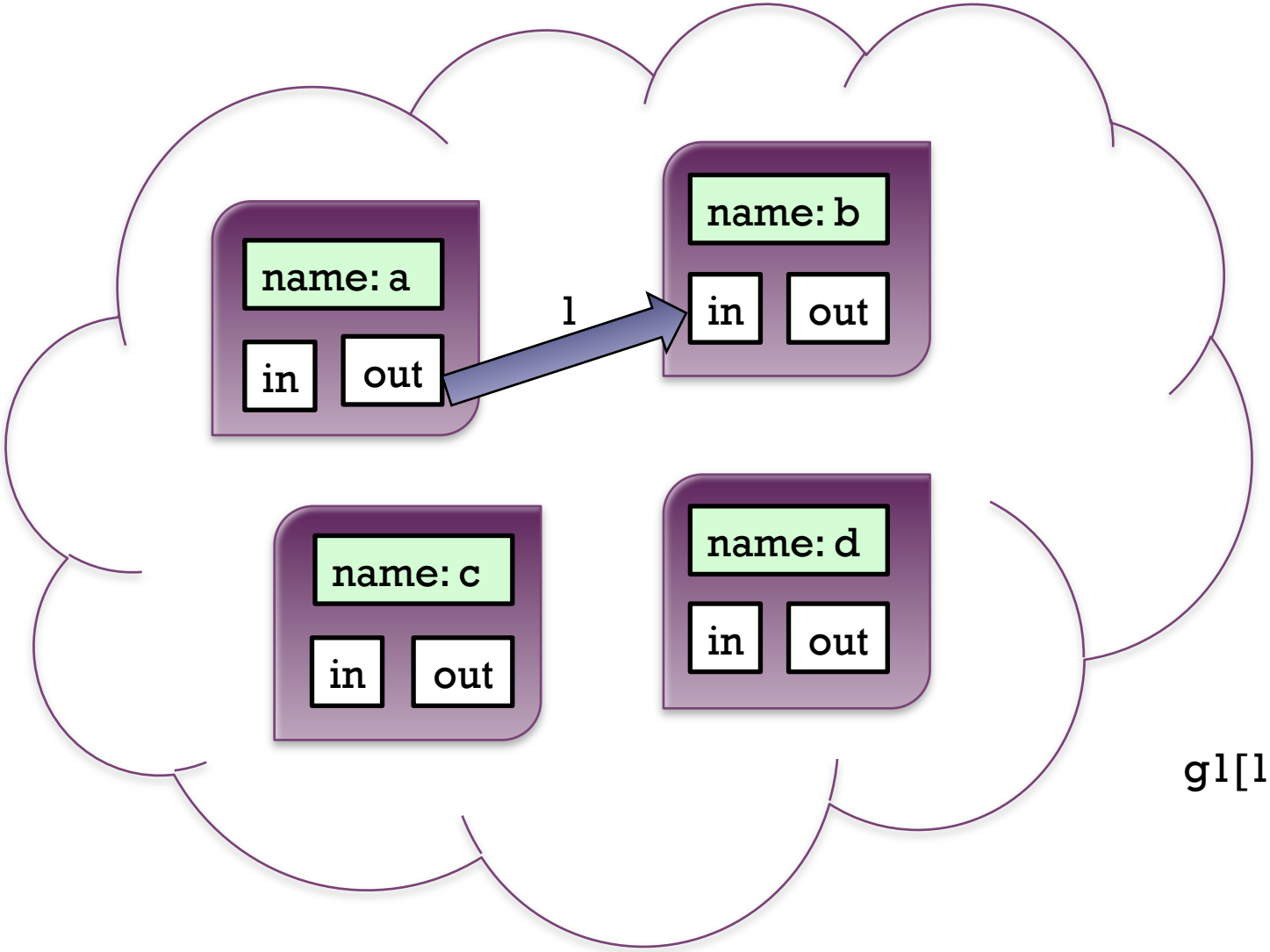

```
a.setData(0);
a.modifyVisited() = false;
print(a@data); /*prints 0*/
prints(a@name); /*prints a*/
```

graph



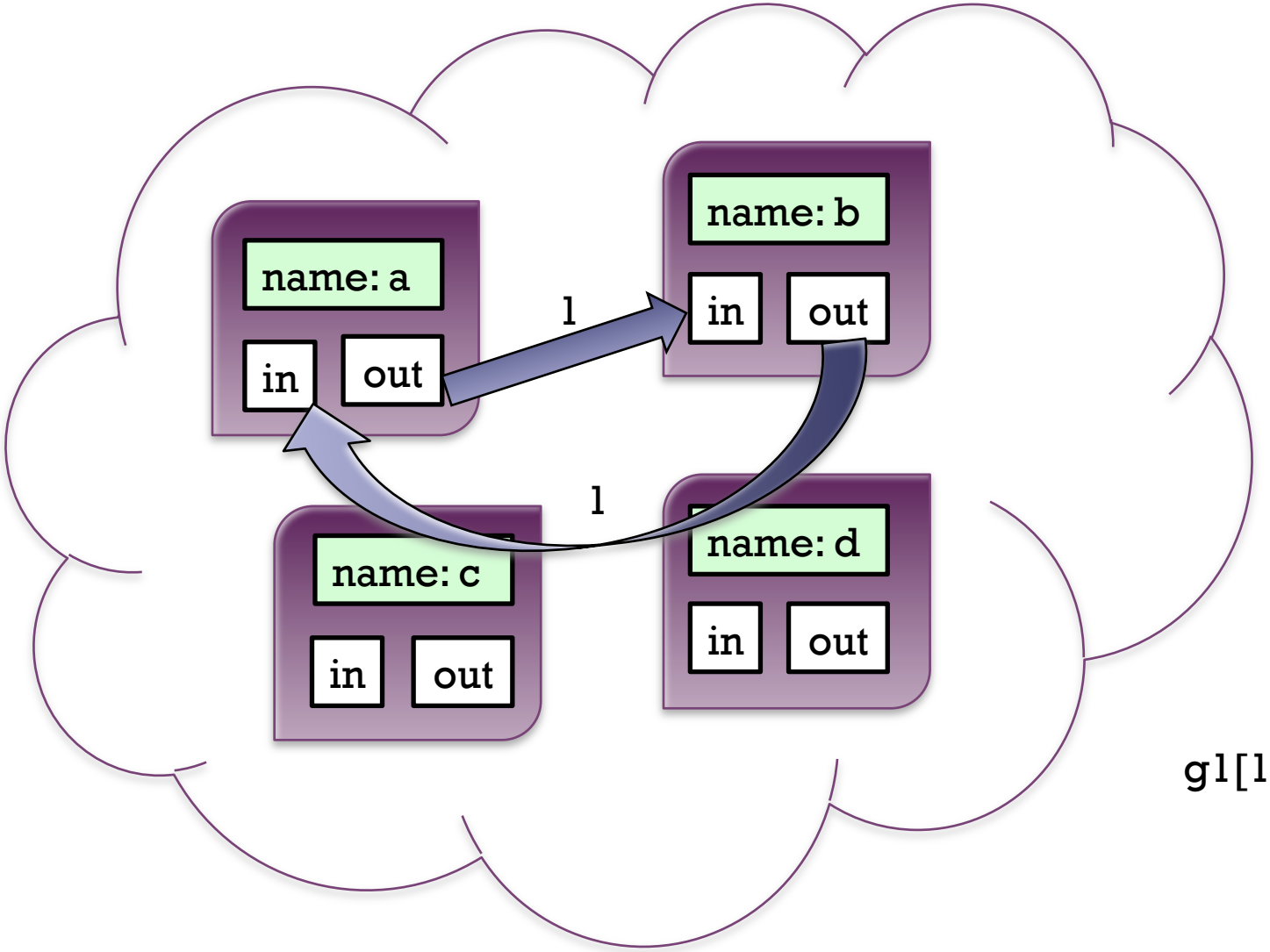

```
gl~+"b";
gl~+"c";
gl~+"d";
...
```

graph



gl[l]->(a, b);

graph



g1[l]->(b, a);

standard library

generic collections

Lists

```
list<int> a;
a = new list<int>(1,2,3);
a.l_add(4);
a.l_delete(0);

int x;
x = a.l_get(0);
print(x);
```

Queues

```
Queue<float> a;
a = new Queue<float>();

a.qadd(3.1);
x = a.qfront();
printfloat(x);
```

+

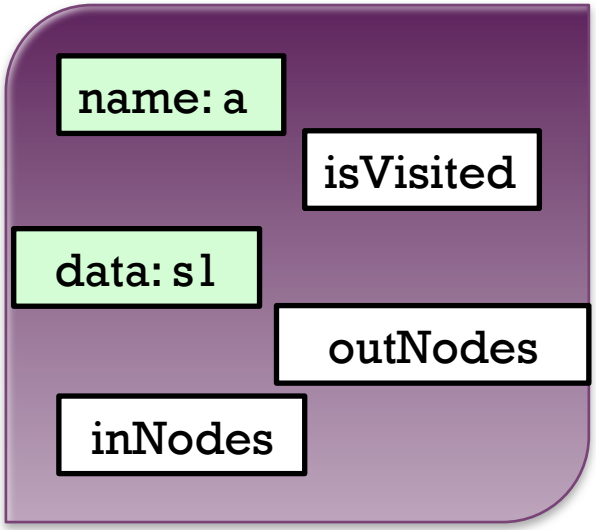
user-defined data types

structs

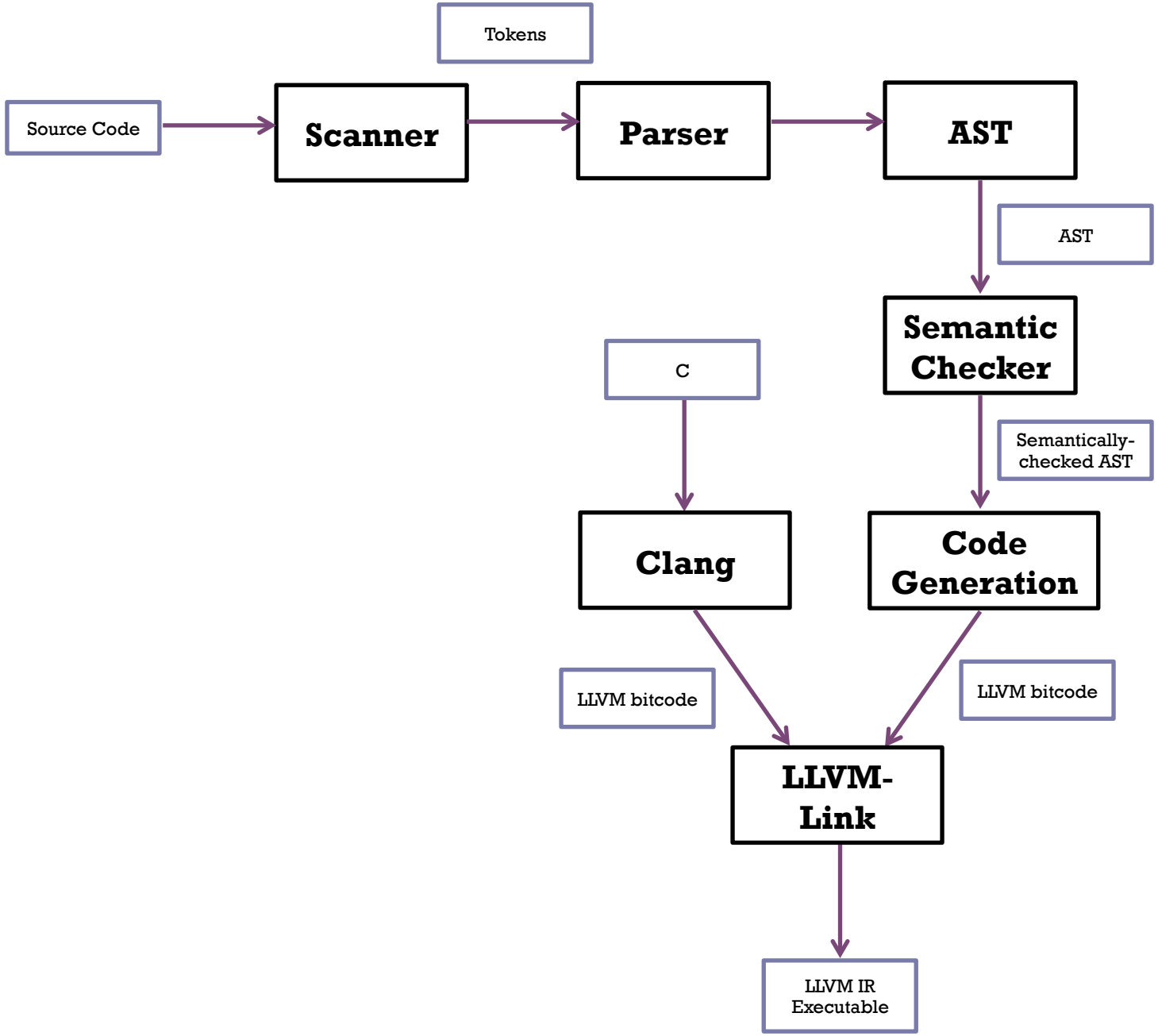
```
struct S1 {
   int x;
   string y;
}

int main() {
   struct S1 s1;
   s1~x = 32;
   s1~y = "hello world";
   return 0;
}
```

structs

```
struct S1 {
   int x;
   string y;
}

int main() {
   struct S1 s1;
   s1~x = 32;
   s1~y = "hello world”;

   graph<string> g1;
   g1 = new graph<string>();
   g1~+"a";
   a = g1~_a;
   a.setData(s1);

   return 0;
}
```



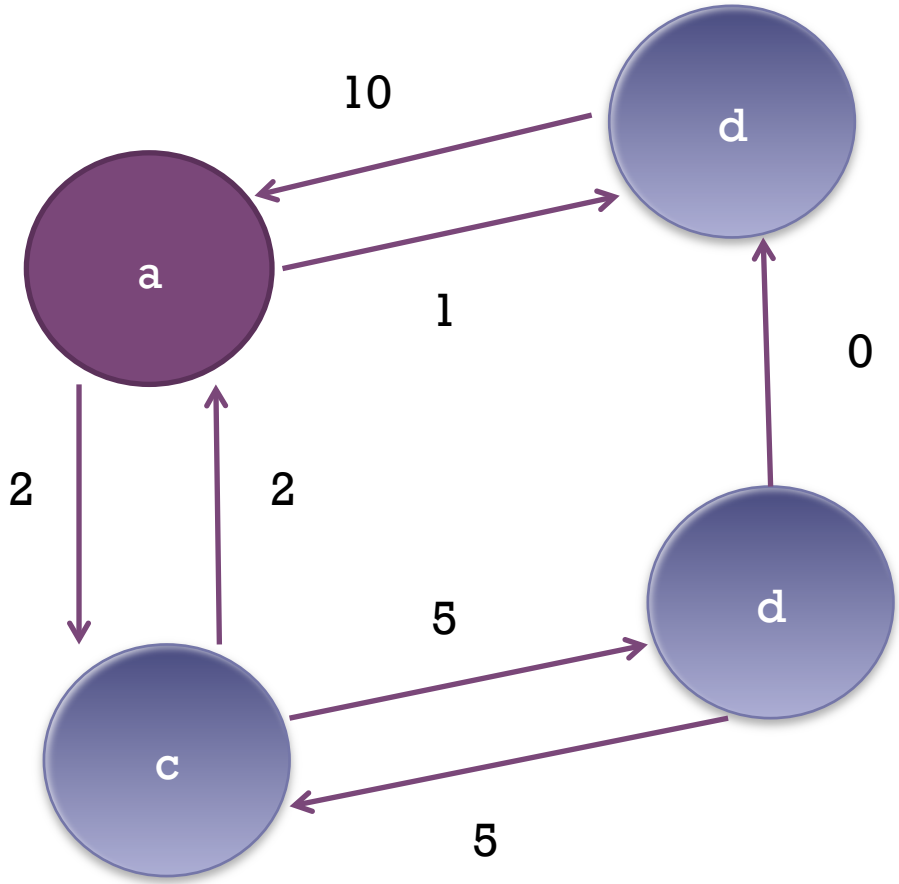
compiler architecture


Tokens
Source Code
Scanner
Parser
AST
AST
Semantic Checker
C
Semantically-checked AST
Clang
Code Generation
LLVM bitcode
LLVM bitcode
LLVM-Link
LLVM IR Executable

+

demo: dijkstra's

dijkstra's



Vertex	Shortest Distance from Source a
a	0
b	1
c	2
d	3