GVL

Graph Visualization Language and Compiler

Programming Language and Translator Fall 2021

Instructor: Prof. Stephen A. Edwards

Team: Minhe Zhang (mz2864)

Yaxin Chen (yc3995)

Aoxue Wei (aw3389)

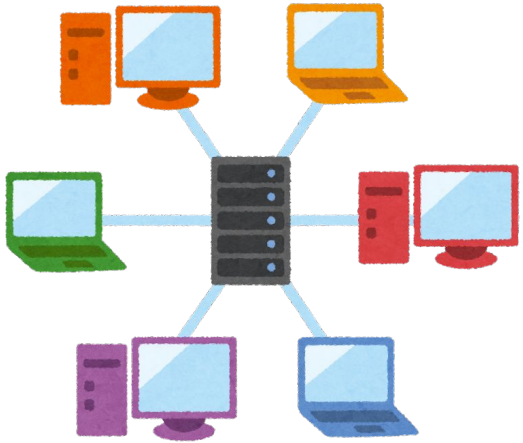
Jiawen Yan (jy3088)

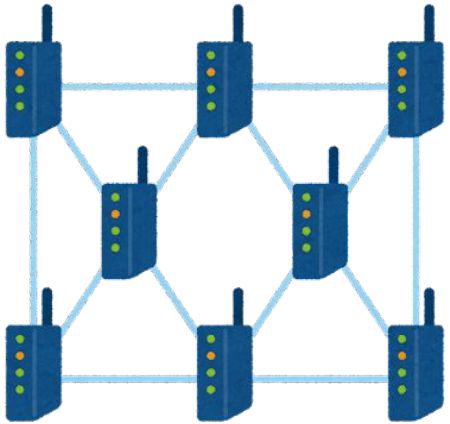
Overview

1. Motivation
2. Environment
3. Compiler Architecture
4. Features
5. Testing
6. Future Work
7. Demo

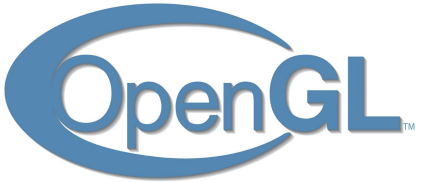
Motivation

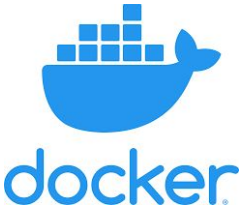
- Easier for beginner to understand
- Convenient for graph algorithm visualization
- The existing tool not good enough



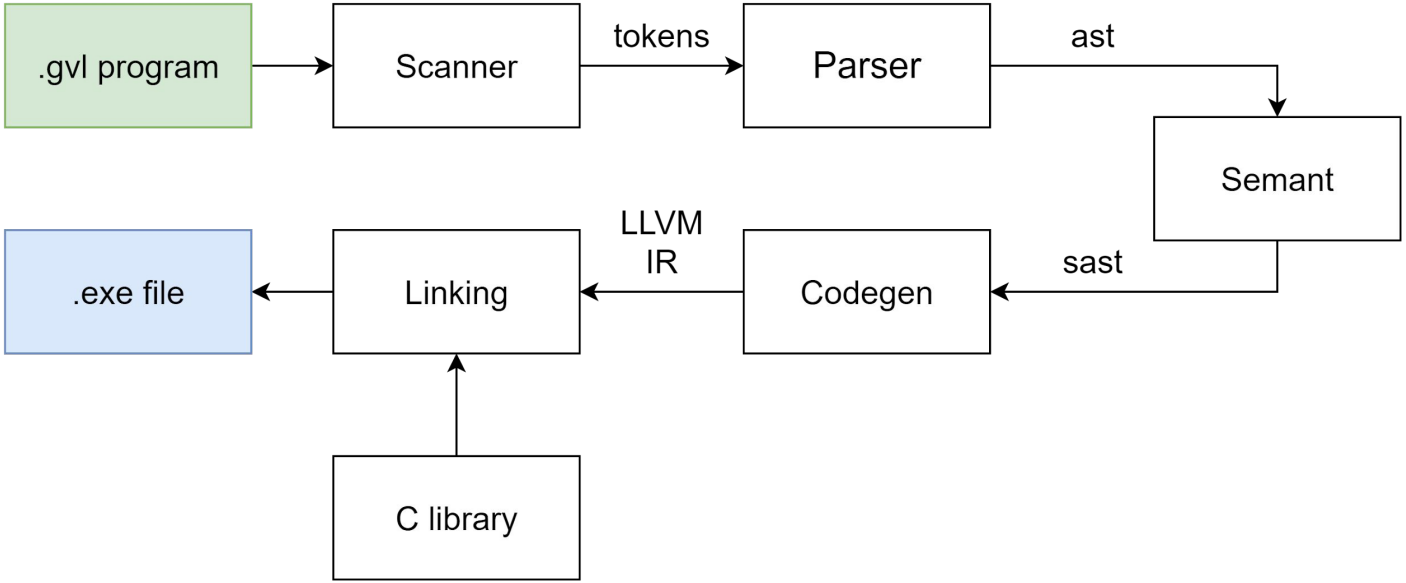


Development Environment





Compiler Architecture



Feature - C-like Syntax

- Imperative
- Statically-typed
- Indentation-insensitive

```
int dfs(graph g, node n) {
    list adjs;
    adjs = get_edges(g, n);

    set_node_color(n, 100, 100, 100);
    printf(get_node_x(n));

    list_iterator iter;
    for (iter = list_begin(adjs); iter != list_end(); iter = list_iter_next(iter)) {
        edge e;
        e = list_iter_data(iter);
        node end_node;
        end_node = get_edge_end(e);
        if (get_node_r(end_node) != 100) {
          dfs(g, end_node);
        }
    }
    return 0;
}
```

Feature - Data Types

- node
- edge
- graph
- list
- list_iterator

```
node n;
edge e;
graph g;

list adjs;
list_iterator li;
```

Feature - Graph

- Set/Get attribute(s) of node, edge, graph
- Get adjacent nodes in graph as a list
- Add/Remove nodes, edges to/from a graph

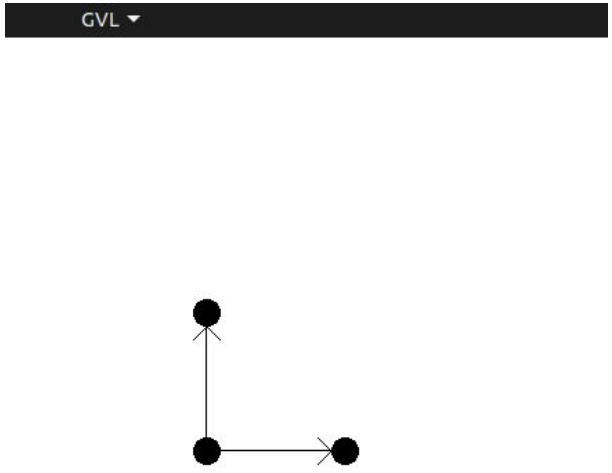
```
int dfs(graph g, node n) {
    list adjs;
    adjs = get_edges(g, n);

    set_node_color(n, 100, 100, 100);
    printf(get_node_x(n));

    list_iterator iter;
    for (iter = list_begin(adjs); iter != list_end();
            iter = list_iter_next(iter)) {
        edge e;
        e = list_iter_data(iter);
        node end_node;
        end_node = get_edge_end(e);
        if (get_node_r(end_node) != 100) {
          dfs(g, end_node);
        }
    }
}
```

Feature - List

- Stores elements of type node/edge
- Supports insertion & removal at both the beginning and end
- Iterable with list_iterator

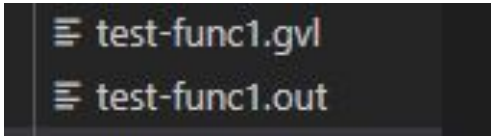
```
int main() {
    list l;
    l = create_list();
    node n1;
    n1 = create_node(100.0, 300.0, 10.0, 0, 0, 0, 0);
    insert_front(l, n1);
    list_iterator l_iter;
    for (l_iter = list_begin(l); l_iter != list_end(); l_iter =
list_iter_next(l_iter)) {
      node n;
      n = list_iter_data(l_iter);
    }
    destroy_list(l);
    return 0;
}
```

Visualization




```
int main() {
    node n1;
    n1 = create_node(150.0, 150.0, 10.0, 0, 0, 0, 0);
    node n2;
    n2 = create_node(150.0, 250.0, 10.0, 0, 0, 0, 0);
    node n3;
    n3 = create_node(250.0, 150.0, 10.0, 0, 0, 0, 0);
    edge e12;
    e12 = create_edge(n1, n2, 1, 0, 0, 0);
    edge e13;
    e13 = create_edge(n1, n3, 1, 0, 0, 0);
    graph g;
    g = create_graph();

    g +o n1;
    g +o n2;
    g +o n3;
    g +- e12;
    g +- e13;

    show_graph(g);

    destroy_graph(g);
    return 0;
}
```

Testing



- Automated testing suite in testall.sh
- Compares output of .gvl program with expected output
  - test-{case}.gvl : Passing tests, output test-{case}.out
  - fail-{case}.gvl : Failing tests, output fail-{case}.err
- If an output is different from the expected output, a .diff file is created

Code Statistic*

Code	C Library	Test
scanner.mll 1100	graph.c 307	functions 76
parser.mly 199	graph.h 101	graph 483
ast.ml 118	graph_visualization.c 147	list 194
sast.ml 79	list.c 178	others 23
semant.ml 217	list.h 42	statements 150
codegen.ml 296	main.c 92	syntax_sugar 39
built-in.ml 101		variable 217
gvl.ml 31		visualization 201
total 1151	**total 867**	**total 1383**

*Number of lines.

Future Work

- User defined payload
- More syntax sugar
- User defined datatype(struct)
- 1-D Array

Demo

Thank you for listening