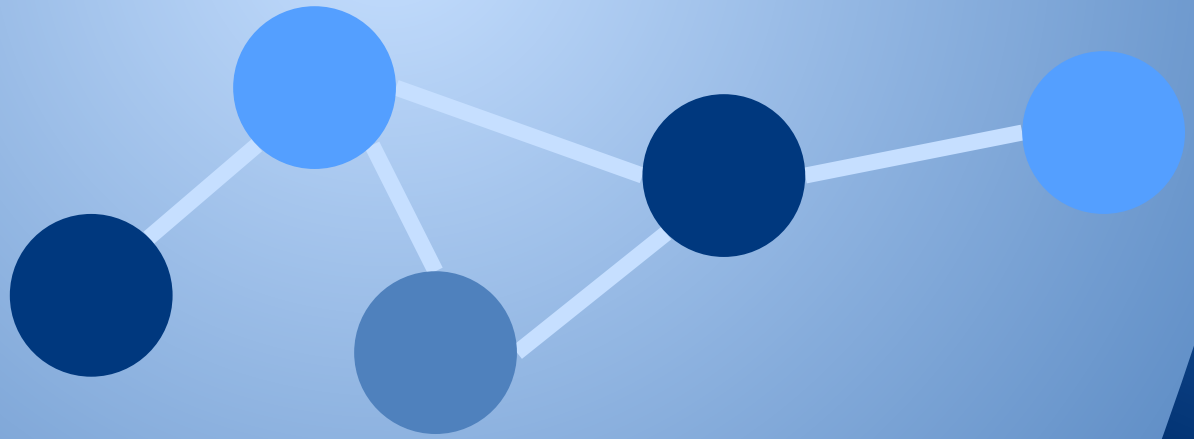
PRTZL

Mathew Mallett mm4673
Rusty Nelson rnn2102
Guanqi Luo gl2483

Why PRTZL?

- Graph programming without GraphDB
- Easy to Use Syntax
- Sandbox Environment



PRTZL Syntax and Features

```
print_string(“hello, world”);
```

```
#include "prtzl.h"
struct graph* g;
int main() {
  g=init_graph();
  print_string(
    "hello, world");
  return 0;
}
```

PRTZL Syntax and Features

```
Number x = 3.14 * 52 + 7;
print_number(x);
String y = “hello “ ^ “world”;
print_string(y);
```

```
#include "prtzl.h"
struct graph* g;
int main() {
  g=init_graph();
  double x = 3.14 * 52 + 7;
  char* y =
    cat("hello ", "world");
  print_number(x);
  print_string(y);
  return 0;
}
```

PRTZL Syntax and Features

```
Vertex a = <+ “a” +>;
Vertex b = <+ “b” +>;
link(a, b, 3.5);
List edges;
edges = a.out;
a.my_property = “value”;
```

```
#include "prtzl.h"
struct graph* g;
int main() {
  g=init_graph();
  struct node* a =
    insert_vertex(g, "a");
  struct node* b =
    insert_vertex(g, "b");
  link(a,b,3.5);
  struct list* edges =
    list_init();
  link(a,b,3.5);
  edges =
    get_node_property(a, "out");
  put_node_property
    (a, "my_property", "value");
  return 0;
}
```

PRTZL Syntax and Features

```
Number x = 1;
if(x == 1)
  print_string(“x == one”);
endif
```

```
#include "prtzl.h"
struct graph* g;
int main() {
  g=init_graph();
  double x = 1;
  if(x == 1){
    print_string("x == 1");
  }
  return 0;
}
```

PRTZL Syntax and Features

```
Number len;
Number i;
List l;
Edge e;
Vertex v = <+"a"+>;
l = a.out;
len = list_length(l);
i = 0;
while(i < len) do
  e = l[i];
  print_edge(e);
  i = i + 1;
endwhile
```

```
#include "prtzl.h"
struct graph* g;
int main() {
  g=init_graph();
  double len = 0.;
  double i = 0.;
  struct list* l = list_init();
  struct node* e;
  struct node* v =
    insert_vertex(g, "a");
  l = get_node_property(v, "out");
  len = list_length(l);;i = 0;
  while(i < len){
    e = list_get(l, i);
    print_edge(e);
    i = i + 1;;
  }
  return 0;
}
```

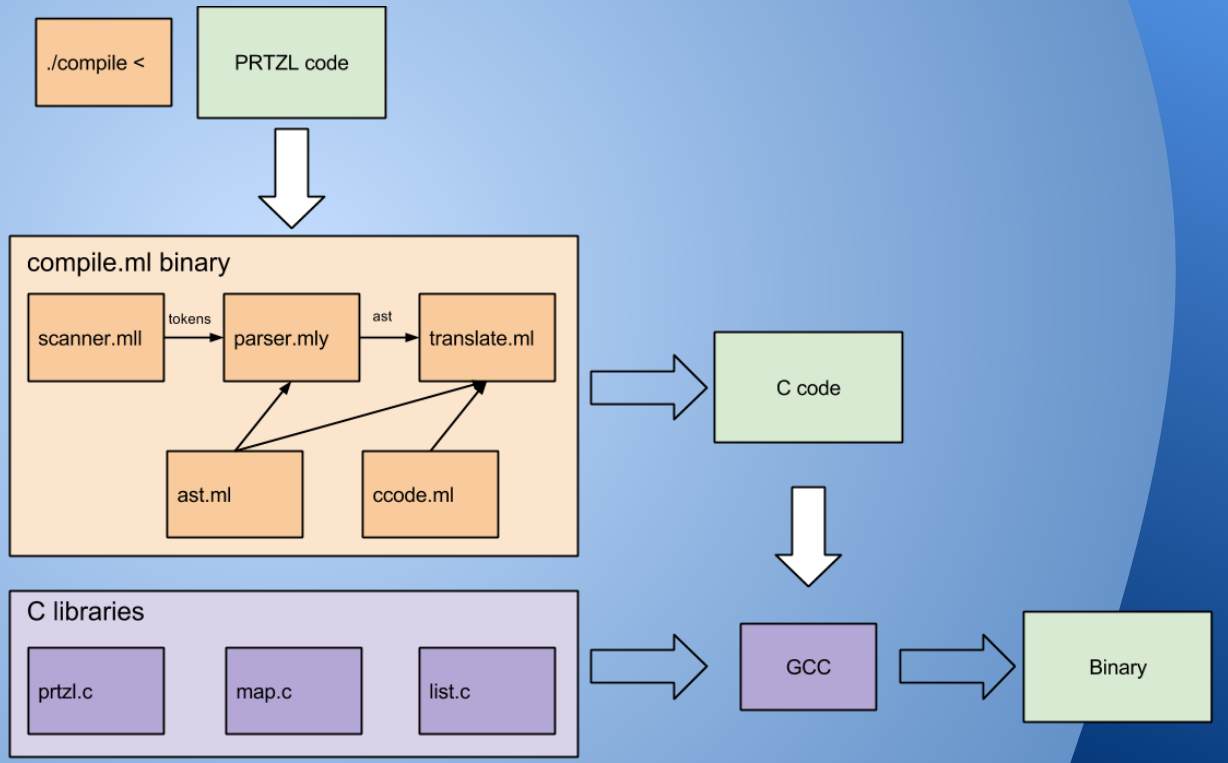
PRTZL Syntax and Features

```
Number dfs(Vertex v)
  List out_list;
  Number i;
  Number len;
  String visited;
  Edge e;
  Vertex dest;
  v.visited = "true";
  print_vertex(v);
  out_list = v.out;
  len = list_length(out_list);
  i = 0;
  while( i < len) do
    e = out_list[i];
    dest = e.dest;
    visited = dest.visited;
    if( !cmp("true", visited) )
      dfs(dest);
    endif
    i = i + 1;
  endwhile
  return 0;
endfunc
```

```
double dfs(struct node* v){
  struct list* out_list = list_init();
  double i = 0.;
  double len = 0.;
  char* visited = "";
  struct node* e;
  struct node* dest;
  put_node_property(v, "visited", "true");
  print_vertex(v);
  out_list = get_node_property(v, "out");
  len = list_length(out_list);
  i = 0;
  while(i < len){
    e = list_get(out_list, i);
    dest = get_node_property(e, "dest");
    visited = get_node_property(dest, "visited");
    if(!(cmp("true",visited))){
      dfs(dest);
    }
    i = i + 1;
  }
  return 0;
}
```

Compiler Architecture

- Lexical Analysis in scanner and parser
- Semantic Analysis & translation in translate.ml
- Library calls translate directly to C library functions



Testing and Development

- Open Source Github Project
- Travis Continuous Integration Testing

```
functions
        Compiling .prtzl        OK
        Compiling .c            OK
        Running .o              OK
link
        Compiling .prtzl        OK
        Compiling .c            OK
        Running .o              OK
strings
        Compiling .prtzl        OK
        Compiling .c            OK
        Running .o              OK
if
        Compiling .prtzl        OK
        Compiling .c            OK
        Running .o              OK
arithmetic_division
        Compiling .prtzl        OK
        Compiling .c            OK
        Running .o              OK
empty
        Compiling .prtzl        OK
        Compiling .c            OK
        Running .o              OK
print
        Compiling .prtzl        OK
        Compiling .c            OK
        Running .o              OK
```

Demo

Looking Ahead

- Enhanced Standard Library
- Multi-File Projects
- Dependency Distribution and Integration Features