A geometric solution language

Qi Wang, Yuechen Zhao
Zichen Chao, Ziyi Luo
Motivation

- Geometry is useful in mathematics, physics, computer science and so many fields.

- But built-in support for graphs are not provided in most programming languages.

- Geo is here to help!

- A simple while powerful language for graph creation and manipulation.

- The best part about Geo - dynamic graphs.
A basic sample

No entry function

```plaintext
function gcd(a: int, b: int): int:

  while(a != b):
    if(a > b):
      a = a - b;
    else:
      b = b - a;
  end

  return a;
end
```

control structures: if-else, while, for

data types: int, float, bool, char, string

keyword end defines the scope
Language Tutorial

- **Geometric types: line, dot, polygon, circle**
  
  dot(x:float, y:float);
  line(dot1:dot, dot2:dot);
  polygons: polygons(num_of_apex:int, apex[]:dot);
  circle: circle(center:dot, radius:float);

- **Presets:**
  
  @panel panelname (essential) - defines a panel
  @end (essential) - the boundary of a specific panel

- **Dynamics:**
  
  model runset: runset(times_of_run:int, g1:geometric_shape, run_para_g1:char, ...);
  function setRunstep(val:float, pos:char):void;
Language Tutorial

Advanced stuff

```plaintext
//panel presets
@panel panel_demo

//geometric shape declaration and initialization
line1 = line(2.0,3.0);
circle1 = circle([3,4], 5);

//runset declaration and initialization
line1.setRunstep(-0.5,'a');
circle1.setRunstep(0.1,'b');
rs = runset(50, line1, 'a', circle1, 'b');

//run statement description
run rs:
set = line1.intersect(circle1);
if (!set.empty())
    print_dot_list(set);
end
@end
```

geometric types:
dot, line, circle and polygons

geometric control
type - runset

keyword run -
dynamic analysis

print intersection
points
Architecture

1. Geo Program
2. Scanner
3. Tokens
4. Parser
5. AST
6. Semantic Check
7. SAST
8. Code Generation
9. Python Code
10. Python Backend
## Architecture

### Source code statistics

<table>
<thead>
<tr>
<th>File</th>
<th>Lines</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>scanner.mll</td>
<td>95</td>
<td>Breaks input stream into tokens</td>
</tr>
<tr>
<td>parser.mly</td>
<td>135</td>
<td>Parses tokens into an AST</td>
</tr>
<tr>
<td>ast.ml</td>
<td>50</td>
<td>Defines acceptable AST structure</td>
</tr>
<tr>
<td>pyast.ml</td>
<td>38</td>
<td>Defines acceptable python AST structure</td>
</tr>
<tr>
<td>compile_sc_py.ml</td>
<td>377</td>
<td>Translates geo AST to python AST</td>
</tr>
<tr>
<td>compile_to_pycode.ml</td>
<td>78</td>
<td>Generates python code</td>
</tr>
<tr>
<td>geo_sc_py.ml</td>
<td>13</td>
<td>Top level</td>
</tr>
</tbody>
</table>
Semantic Check

- Use StringMap to implement translation environments
  - vars: keep information about variables
  - funcs: keep information about functions
  - func_opt: keep information about types of function parameters

- Check for:
  - undeclared variables and functions
  - mismatched types
  - wrong types
  - function parameters not match
  - undefined operations
  - Geo syntax error
  - .....

Code Generation

Algorithm Example (demo_fb.g)

```plaintext
@panel gcd
function gcd(a:int, b:int):int:
  if (a<b):
    return (gcd(b,a));
  else:
    if (a == b):
      return (a);
    else:
      return(gcd(a-b, b));
  end
end
print(gcd(70,28));
print(gcd(147,21));
@end
```

```plaintext
from Tkinter import *
from sysgeo import *
def gcd(a, b):
  if (a < b):
    return gcd(b, a)
  else:
    if (a == b):
      return a
    else:
      return gcd((a - b), b)
PI = 3.14159265359
print gcd(70, 28)
print gcd(147, 21)
```
Code Generation

**Algorithm Example**

```plaintext
@panel qsort
function qsort(a:list, l:int, r:int):list:
    i = l; j = r; mid = (l+r)/2;
    while (i <= j):
        while (i <= j & a[i] < a[mid]):
            i = i+1;
        end
        while (i <= j & a[j] > a[mid]):
            j = j-1;
        end
        if (i <= j):
            k = a[i]; a[i] = a[j]; a[j] = k; i = i+1; j = j-1;
        end
    end
    if (l < j):
        a = qsort(a, l, j);
    end
    if (i < r):
        a = qsort(a, i, r);
    end
    return(a);
end

b = {3,7,8,32,1,4,7,9,2,5}; b = qsort(b, 0, len(b)-1); print(b);
@end
```
Code Generation

- Graph Example

```python
@panel panel1

c1 = circle([0, 0], 2);
c2 = circle([2, 0], 2);
l1 = line([2,-4],[2,4],0,0);
l2 = line([-2,0],[6,0],[-2,6]);
r1=runset(360,0.05);
r1.addPara(c1,'r');
r1.addPara(c2,'r');
r1.addPara(l1,'b');
r1.addPara(l2,'b');

run r1:
    p1 = c1.getPointbyarc(r1.getRuncount()*PI/(-36));
    print(p1);
    l1.rotateonPoint(p1,PI/36);
    l2.rotateonPoint(p1,PI/36);
    c2.setCenter(p1);
    t1=l1.getEndpoints();
    r1.mark(t1[0]);
    r1.mark(t1[1]);
    t1=l2.getEndpoints();
    r1.mark(t1[0]);
    r1.mark(t1[1]);

end
@end
```
## Testing

### Test case statistics – comprehensive check

<table>
<thead>
<tr>
<th>File</th>
<th>Lines</th>
<th>File</th>
<th>Lines</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>test-assignments.g</td>
<td>14</td>
<td>test-assignments.ref</td>
<td>6</td>
<td>All kinds of assignments</td>
</tr>
<tr>
<td>test-circle.g</td>
<td>19</td>
<td>test-circle.ref</td>
<td>12</td>
<td>Geo type circle &amp; obj funcs</td>
</tr>
<tr>
<td>test-comparison.g</td>
<td>11</td>
<td>test-comparison.ref</td>
<td>6</td>
<td>Comparison &amp; boolean opts</td>
</tr>
<tr>
<td>test-dot.g</td>
<td>7</td>
<td>test-dot.ref</td>
<td>3</td>
<td>Geo type dot &amp; obj funcs</td>
</tr>
<tr>
<td>test-fib.g</td>
<td>18</td>
<td>test-fib.ref</td>
<td>9</td>
<td>Recursive function</td>
</tr>
<tr>
<td>test-for.g</td>
<td>5</td>
<td>test-for.ref</td>
<td>10</td>
<td>For statements</td>
</tr>
<tr>
<td>test-function.g</td>
<td>37</td>
<td>test-function.ref</td>
<td>1</td>
<td>Function &amp; if &amp; while</td>
</tr>
<tr>
<td>test-gcd.g</td>
<td>15</td>
<td>test-gcd.ref</td>
<td>2</td>
<td>Function &amp; if statement</td>
</tr>
<tr>
<td>test-if.g</td>
<td>11</td>
<td>test-if.ref</td>
<td>1</td>
<td>If statements (nested)</td>
</tr>
<tr>
<td>test-line.g</td>
<td>35</td>
<td>test-line.ref</td>
<td>15</td>
<td>Geo type line &amp; obj funcs</td>
</tr>
<tr>
<td>test-list.g</td>
<td>8</td>
<td>test-list.ref</td>
<td>4</td>
<td>List</td>
</tr>
<tr>
<td>test-operations.g</td>
<td>20</td>
<td>test-operations.ref</td>
<td>9</td>
<td>Check +-*/%^% operatations</td>
</tr>
<tr>
<td>test-polygon.g</td>
<td>20</td>
<td>test-polygon.ref</td>
<td>11</td>
<td>Geo type polygon &amp; obj funcs</td>
</tr>
<tr>
<td>test-print.g</td>
<td>10</td>
<td>test-print.ref</td>
<td>8</td>
<td>Print function</td>
</tr>
<tr>
<td>test-qsort.g</td>
<td>35</td>
<td>test-qsort.ref</td>
<td>1</td>
<td>List &amp; recursive function</td>
</tr>
<tr>
<td>test-while.g</td>
<td>7</td>
<td>test-while.ref</td>
<td>6</td>
<td>While statement</td>
</tr>
</tbody>
</table>
## Test case statistics – error check

<table>
<thead>
<tr>
<th>File Name</th>
<th>Lines</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>error-semantics1.g</td>
<td>2</td>
<td>Undeclared function</td>
</tr>
<tr>
<td>error-semantics2.g</td>
<td>5</td>
<td>Function input parameter type error</td>
</tr>
<tr>
<td>error-semantics3.g</td>
<td>6</td>
<td>Undefined argument</td>
</tr>
<tr>
<td>error-semantics4.g</td>
<td>3</td>
<td>Char cannot plus int/float</td>
</tr>
<tr>
<td>error-syntax1.g</td>
<td>1</td>
<td>Lose end</td>
</tr>
<tr>
<td>error-syntax2.g</td>
<td>1</td>
<td>Unrecognized token</td>
</tr>
<tr>
<td>error-syntax3.g</td>
<td>1</td>
<td>Lose semicolon</td>
</tr>
<tr>
<td>error-syntax4.g</td>
<td>2</td>
<td>Wrong function declaration</td>
</tr>
<tr>
<td>error-syntax4.g</td>
<td>1</td>
<td>If statement error</td>
</tr>
</tbody>
</table>
Testing

Auto check – geotestall.sh

First: Check whether all files can be successfully compiled;
Then: Compared the output with the ref answer.

Compiling tests/test-assignments.g...
Compiling tests/test-circle.g...
Compiling tests/test-comparison.g...
Compiling tests/test-dot.g...
Compiling tests/test-fib.g...
Compiling tests/test-for.g...
Compiling tests/test-function.g...
Compiling tests/test-gcd.g...
Compiling tests/test-if.g...
Compiling tests/test-line.g...
Compiling tests/test-list.g...
Compiling tests/test-operations.g...
Compiling tests/test-polygon.g...
Compiling tests/test-print.g...
Compiling tests/test-qsort.g...
Compiling tests/test-while.g...
diff -b tests/test-print.out tests/test-print.ref > tests/test-print.diff
diff -b tests/test-qsort.out tests/test-qsort.ref > tests/test-qsort.diff
diff -b tests/test-while.out tests/test-while.ref > tests/test-while.diff
OK
###### SUCCESS
Lessons Learned

■ Qi Wang:
  “Start early on the project and make a plan ahead, if things are different from scheduled, discuss together and activate soon.”

■ Yuechen Zhao:
  “Effective communications are the key to success, do not waste too much time on arguing plans, but discussion is important.”

■ Zichen Chao:
  “Keep the whole picture in mind, modify the plan as the project progressed and learn Ocaml as early as possible!”

■ Ziyi Luo:
  “Comprehensive test cases are important and test early, you can never imagine how many problems you may encounter when testing.”