dots

A Graph Language

Hosanna Fuller (hjf2106) — Manager
Yumeng Liao (yl2908) — Tester
Adam Incera (aji2112) — C Guru
Rachel Gordon (rcg2130) — Language Guru

December 2015

Contents

1 Introduction

Graphs are a powerful and versatile data structure used across many languages to help visually organize and manipulate data. Many languages do not provide out-of-the-box support for data structures and methods useful in solving graph problems. Because of this, programmers end up spending unnecessary time and energy implementing these critical structures themselves. As a result, graph implementations often widely vary in efficiency and modularity. The goal of d.o.t.s. is to provide an out-of-the-box graph framework so that users can focus on creating the algorithms needed to solve their problems, rather than getting bogged down in implementation issues. With d.o.t.s. users can comprehensively solve a wide variety of graph-based problems. Some example problems include: expressing network relationships such as a series of interconnected routers with edge costs, representing decision trees in probability, and running analyses on propositional models.

1.1 Summary

d.o.t.s. is a scripting language, much like python, with the goal to take the headache out of implementing and manipulating graphs and their data. Many fields of study in computer science such as Network Routing and Natural Language Processing depend on graphs to solve complex algorithmic issues. As a result, programmers spend unnecessarily long amounts of time just building up the graph structure they need to represent their network or language heuristic. Similarly, even when they do have their graph framework implemented, operations for manipulating the data are often syntactically confusing or take several lines of code to do a simple operation. For exmample adding a node to a graph may entail scanning the entire graph for duplicates, creating the new node, and appending it to the end of a list. Our hope with d.o.t.s. is that programmers find it easy to use so they can focus on the real algorithmic problems they are trying to solve instead of getting bogged down in implementation details.

1.2 Use Cases

Instead of spending hours implementing standard data structures like graphs, programmers can focus on implementing complex algorithms. d.o.t.s. offers users the opportunity to work with graphs in a simple, intuitive manner. Here are some example use cases below:

1. Build graphs to help visualize data. The graph structure allows users to easily create nodes and assign them data. Similarly you can iterate through all the graphs data easily so displaying your graph is as simple as writing a pretty printer.
2. Social Network Analysis – For instance, if you wanted to have a program that stored a graphical representation of all your Facebook friends you could easily create that graph and simply add or subtract friends from that graph without ever having to write a custom function to do so. d.o.t.s. allows users to maintain graph structures for storing data easily.
3. Graph Algorithms / Routing Algorithms – Users can quickly implement complex graphing algorithms like Dijkstra's with d.o.t.s.. We were able to create Dijkstra's example in 30 lines of code. If you were to try and implement Dijkstra's in java it would be at least 500 lines.
4. Propsitional Models – propositional models can be represented as a graph, where edge weights are probabilities conditioned on the parent nodes. Using d.o.t.s., a programmer could run statistical analyses on a graph.

2 Setup

The following are instructions for getting the d.o.t.s. compiler up and running.

2.1 Installation

Note: all commands are run from the `src` directory of the compiler.

1. Unpack the d.o.t.s. compiler tar ball.
2. Add the path to the `clib` directory to the `DOTS` environment variable.

   ```
   > export DOTS=/path/to/dots-compiler/src/clib
   ```
3. Make the C library.

   ```
   > make setup
   ```
4. Make the d.o.t.s. compiler.

   ```
   > make
   ```

2.2 Running the compiler

Compiling d.o.t.s. code consists of two pieces: compiling the `.dots` source file into a `.c` file, and compiling the `.c` file into a binary executable.

Both of the these two steps are wrapped up in the `gdc` executable file to allow for easy compilation of d.o.t.s. programs. The script takes two arguments: the `.dots` file to compile, and the name of the binary file to compile to. The first argument is required, the second argument is optional.

Example:

Compile the file to the default executable – `exec`

```
> ./gdc dots/src/sample-code/sample01.dots
> ./exec
```

Compile the file to a specific executable name

```
> ./gdc dots/src/sample-code/sample01.dots example-file
> ./example-file
```

3 Basic d.o.t.s. Tutorial

In this section, we walk you through creating your first d.o.t.s. program.

Step 1: Creating Your d.o.t.s. source file

Create a new file called `tutorial.dots` in any directory, and open it up.

Step 2: Declaring Primitive Variables

In d.o.t.s., you declare variables by writing a datatype, followed by an alphanumeric string that begins with an alphabetic letter. The basic data types are: `num, string, bool`.

```
num x;
string s;
```

Step 3: Assigning Variables

Now let's give each of those variables a value.

```
x = 5.3;
s = "This is a d.o.t.s. program.";
```

Step 4: Declaring Nodes

The basic element of a graphs in d.o.t.s. are objects of the type: `node`. When declaring a node, you have the option of assigning the node's value with a string in parentheses.

```
node n1;
node n2("florida");
```

Step 5: Declaring and Assigning Graphs

The main data type underlying d.o.t.s. is the `graph` type. A graph represents a collection of node objects. Graphs can be assigned to an expression that evaluates to a graph object. Let's create a graph consisting of the two nodes we just declared.

```
graph g;
g = n1 + n2;
```

Step 6: Assigning Edges and Weights

Now we have a graph consisting of the two nodes `n1` and `n2`. But neither of these nodes have incoming or outgoing edges. Let's spice up the graph a little by adding in some edges.

```
n1 -->[2.3] n2;
n2 --> n1;
```

Now `n1` has an edge leading to `n2` with weight 2.3, and `n2` has an edge leading to `n1` with weight 0.

Step 7: Declaring Complex Variables

The collection data types are: `list<elem_type>` and `dict<key_type, val_type>`, where `elem_type`, `key_type`, and `val_type` are all d.o.t.s. data types. These data types are used to keep track of multiple values at once.

Lists can be declared and assigned on the same line, and can be assigned to list literals. Let's create a list of our nodes.

```
list<node> node_list = [n1, n2];
```

Currently, `dict` literals are not supported, so let's create a dict and assign each element separately.

```
dict<string, num> lang_prob;
lang_prob["e"] = 12.7;
lang_prob["a"] = 8.17;
lang_prob["d"] = 4.25;
```

Step 8: Printing Results

Now lets print out some of what we've done. d.o.t.s. has a `print` function that can take a comma-separated list of values of any data type[1]. To print out the nodes inside the graph we created, we'll use a `for` loop to iterate over the graph's nodes.

```
print(x, "\n", s, "\n", n1, "\n", n2, "\n");

for (elem in g) {
   print(elem, "\n");
}

print(node_list, "\n", lang_prob, "\n");
```

Step 9: Putting It All Together

Your `tutorial.dots` file should now look like this:

```
num x;
string s;

x = 5.3;
s = "This is a d.o.t.s. program.";

node n1;
node n2("florida");

graph g;
g = n1 + n2;

n1 -->[2.3] n2;
n2 --> n1;

list<node> node_list = [n1, n2];
dict<string, num> lang_prob;
lang_prob["e"] = 12.7;
lang_prob["a"] = 8.17;
lang_prob["d"] = 4.25;

print(x, "\n", s, "\n", n1, "\n", n2, "\n");

for (elem in g) {
   print(elem, "\n");
}

print(node_list, "\n", lang_prob, "\n");
```

Listing 1: tutorial.dots

Step 10: Compiling tutorial.dots

Note: this step assumes you've already done the installation described in Section 2.1

[1]directly printing graph objects not yet supported

Navigate to your `/path/to/dots/compiler/src` directory. In your terminal enter:

```
> ./gdc /path/to/tutorial.dots
```

You should now see a file named `exec` in the `src` directory.[2] Run it by doing:

```
> ./exec
```

Your output should look something like this:

```
5.300
This is a d.o.t.s. program.
N--633324960("")
N--633324896("florida")
N--633324960("")
N--633324896("florida")
[N--633324960(""), N--633324896("florida"), ]
{a: 8.170, d: 4.250, e: 12.700, }
```

4 Language Reference Manual

4.1 Introduction

The strict typing and control flow in d.o.t.s. is reminiscent of C and Java, but overall the language is intended to be used more as a scripting language, where the user builds their graphs quickly using the intuitive node and edge operators and then performs operations on the structures.

The d.o.t.s. compiler compiles code written in d.o.t.s. into C, and then uses the GNU C Compiler to build binary executables.

Note: In the following sections, the word "graph" is sometimes used to denote a data structure and sometimes to denote the abstract structure from computer science and mathematics:

> A graph data structure consists of a finite (and possibly mutable) set of nodes or vertices, together with a set of ordered pairs of these nodes (or, in some cases, a set of unordered pairs). These pairs are known as edges or arcs. As in mathematics, an edge (x,y) is said to point or go from x to y. The nodes may be part of the graph structure, or may be external entities represented by integer indices or references.
>
> A graph data structure may also associate to each edge some edge value, such as a symbolic label or a numeric attribute (cost, capacity, length, etc.). [3]

For the sake of clarity, from this point forward we will refer to the language-specific data structure using the lowercase "graph" and the mathematical concept using the uppercase "Graph."

4.2 Lexical Conventions

4.2.1 Comments

A comment is a sequence of characters ignored by the compiler in a d.o.t.s. script.

[2] You can see the C code that's code that's compiled in Appendix E.4

[3] `https://en.wikipedia.org/wiki/Graph_(abstract_data_type)`

Syntax	Comment Style
* *code* *\	multi-line comment
#	single-line comment

Table 1: Comment Styles

4.2.2 Identifiers

Variable and function names must begin with an alphabetic character and contain only alphanumeric characters.

4.3 Keywords

Table 2 shows the list of reserved words in d.o.t.s.. Keywords have special meaning in d.o.t.s. and cannot be overwritten.

if
else
true
false
in
for
while
return
bool
num
string
node
graph
list
dict

Table 2: Keywords

4.4 Built-in Data Types

d.o.t.s. provides three different types of variables: basic types, built-in types, and collections. The three basic types are `num` `bool` `string`. Each of these basic types have raw value representations, which can be used with no prior declaration of variables. Such values can also be assigned as the values of variables.

The two built-in data types, `node` and `graph`, provide the basis for algorithms written in d.o.t.s.. The built-in collection types are `list` and `dict`, and can be used to contain any of the other two kinds of types.

d.o.t.s. is a strictly-typed language, meaning that the types of all variables must be declared at the same time that the variable is declared.

Data Type	Fields
num	
string	
bool	
list	
dict	
node	value, in, out
graph	vertices

Table 3: Built-in Data Types

4.4.1 Nums and Strings

Category	Data Type	Operator	Explanation
comparison	num, string	<, >, <=, >=, !=, ==	Operate in the same way as languages such as C/C++, with the exception that string equality compares the *value* contained in the string.
computation	num	+, -, *, /, %	Operate in the same way as languages such as C/C++.
concatenation	string	+	String concatenation operator

Table 4: Num and String Operators

Both nums and strings can be assigned on the same line as their declaration.

In addition to literal `num` values, d.o.t.s. provides an infinity value for the `num` type: `INF`. The operators perform a little differently for these values.[4] As the primary use of infinity in graph problems is to define edge weight and not to perform mathematical calculations, the computation operators cause an exception whenever infinity is an operand.

For comparison operators, `INF` is greater than all non-null non-infinity values and equal to other infinity values. Defining the comparison operators for `INF` values allows them to be used as valid edge weights, which can be useful for graph problems.

```
num x = 12.0;
num y = 6 * 2.0;
num z = 6.5;
x == y; # returns true
y < z;  # returns false


string phrase = "Hello" + " " + "World"; # phrase = "Hello World"
string a = "cat";
string b = "bear";
a < b;  # returns true;

num i = INF;
y > INF; # returns false
num h = 2 + i; # exception
```

Listing 2: Demonstration of “num” and “string” types.

[4]Note: We didn’t have time to implement special handling of INF.

4.4.2 Node and Graph Objects

The data types which underpin d.o.t.s. and give it its advantage in the Graph domain over languages such as C are `node` and `graph`. From the get-go any programmer using d.o.t.s. can use these data types to quickly build Graphs without the need to waste time creating these data structures from scratch.

A `node` object represents a single vertex in a Graph, whereas a `graph` object represents a collection of graphs (which can be empty). Conceptually: A node is a graph, but a graph is not a node.

Recursive definition of graph objects:

- An empty graph is a graph.
- A node is a graph.
- A graph added to a graph is a graph.

A graph contains only the field `nodes`, which is a list of all node objects contained within the graph.

A node contains the fields `val, in, out`. Internally, a node is uniquely identified by its address in memory, but this distinction is visible to the programmer only when they print a node. The `val` field is a string object, and simply represents some value that the node contains. One possible use of the value field is to allow users to assign a more semantic meaning to nodes (ex. setting the value to the name of a city). The `in` field is a dict mapping nodes that the current node has edges into to weights. Similarly, the `out` field is a dict mapping nodes that have edges into the current node to weights. The keys of the two dicts are nodes. The `in` and `out` fields of a node can only be accessed by calling the `ine()` and `oute()` member functions of node. An example of accessing the `in` and `out` dicts of a node can be seen in Listing 5.

Nodes can be declared in two different ways. In the first, the variable can simply be declared with the `node` keyword and a variable name. This creates a basic node with an empty value, `in` list, and `out` list. In the second manner, a node can be declared by giving it an initial value inside parentheses after the variable name (as seen in Listing 3).

Graphs can be declared with the keyword `graph`. There is also a special graph-creation syntax that can *only* be used at declaration time of a graph object (as seen in Listing 3). This special syntax consists of a comma-separated list of node operations (an example of this syntax can be found in Listing 5). Each node referenced in this type of assignment must have been previously declared. This special syntax adds the nodes within the declaration to the declared graph and adds the specified edges to each of the nodes.

After declaration, as with all other types, nodes and graphs can be assigned any expression that evaluates to something of the same type.

```
node x;
node y("nyc");

graph g1;
graph g2;
g2 = g1;

graph g3 = {
   x -->[2.3] y,
   y --> x
}; # special graph assignment syntax only available at declaration time
```

Listing 3: Declaration of “node” and “graph” objects.

4.4.3 Collections – Dicts and Lists

Lists are declared using the keyword `list` and an indicator of the type of the list, as all objects in a list must be of the same type. Lists can be assigned by putting a comma-separated list of objects inside brackets, or can be filled using list functions. Lists are allowed to be assigned at declaration time.

Dictionary objects in d.o.t.s. represent mappings from objects to objects; a key can be any type for which the comparison operator "==" has been defined, and all keys in a given dict must be of the same type. Similarly, all values in a given dict must be of the same type. Dict objects are declared in a similar manner to lists, using the keyword `dict` and an indicator of the type that the dict maps from and to. Dicts must be assigned after declaration, and can be assigned by putting a comma-separated list of `key:value` pairs inside curly braces.

Operation	Syntax	Explanation
List Ops		
random access	`listVar[index]`	returns the element at the index
enqueue	`listvar.enqueue(elemVar)`	appends the given variable to the end of the list
dequeue	`listvar.dequeue()`	removes and returns the front element of the list
push	`listvar.enqueue(elemVar)`	appends the given variable to the front of the list
pop	`listvar.dequeue()`	removes and returns the front element of the list
dequeue	`listvar.dequeue()`	removes and returns the front element of the list
peek	`listvar.enqueue(elemVar)`	returns the variable at the front of the list
Dict Ops		
value access	`dictVar[key]`	returns the value at the given key. *key* can be a raw value or variable.
value assignment	`dictVar[key] = newValue`	if *key* already exists in the dict, overrides the old value with *newValue*, else adds the (*key,newValue*) pair to the dict.
element deletion	`dictVar.remove(keyVal)`	removes the (key, value) pair with key *keyVal* from the dict.

Table 5: List and Dict Operations

```
list<num> numList = [1, 2, 3];

list<string> strList = ["Hello"];
strList.enqueue(" ");
strList.enqueue("World"); # strList now equals ["Hello", " ", "World"];
string s = strList.peek(); # s = "World"
strList.dequeue(); # strList = ["Hello", " "]

dict<string, num> numDict;
numDict["one"] = 1;
numDict["two"] = 2;
numDict["three"] = 3;
```

```
numDict.remove("three");
/* Now:
   numDict = {"one" : 1, "two" : 2}
   success = true
 */

dict<node, num> cityRankings;
node x("chicago");
cityRankings[x] = 2;
node y("seattle");
cityRankings[y] = 1;
/* Now:
   cityRankings = {x : 2, y : 1}
 */
```

Listing 4: Declaration of "list" and "dict" types.

4.5 Special Operators

4.5.1 Node Operators

The node operators outlined in Table 6 [5] are all binary operators which take a node object on the left-hand and right-hand sides of the operator.

Operator	Explanation
`--`	Add undirected edge with no weights between the 2
`-->`	Add directed edge from left node to right node with no weight
`-->[`*num*`]`	Add a directed edge from the left node to the right node with weight *num*
==	Returns whether the internal ids of 2 nodes match
!=	Returns whether the internal ids of 2 nodes do not match

Table 6: Node Operators

[5] Note: Weighted undirected edges (ex. `x --[7] y`) are buggy in final version

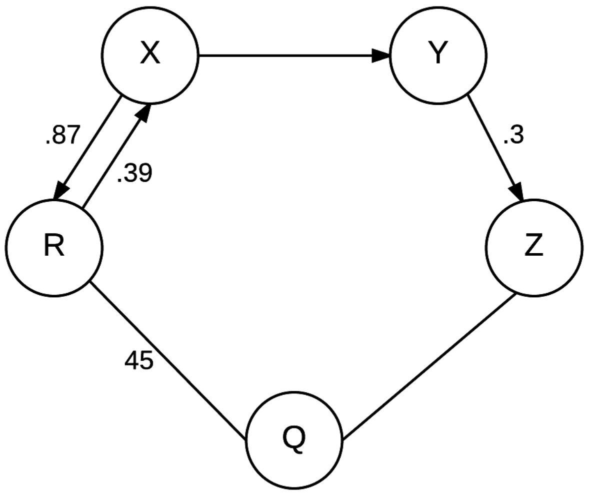


Figure 1: Example Graph showing nodes with different weights and edges.

```
node X;
node Y;
node Z;
node Q;
node R;

X --> Y;
Y -->[.3] Z;
Z -- Q;
R [.87]--[.39] X;
# and since Q --[45] R is buggy, use:
Q -->[45] R;
R -->[45] Q;

node temp;
temp = R;

R == Q;  # returns false
R == temp; # returns true

/* accessing edge lists: */
X.oute()[Y]; # == null
Y.oute()[Z]; # == .3
R.ine()[X]; # == .87

node alt = Z;
Y.oute()[alt] == Y.oute()[Z]; # returns true

/* Alternate Graph Creation:
 * adds the nodes to the graph G, while
 * at the same time it adds edges and weights
 * between the nodes
 */
node x, y, z, q, r;
graph G = {
   x --> y,
   y -->[.3] z,
   z -- q,
   q --[45] r,
   r [.87]--[.39] x
};
```

Listing 5: Shows the use of node operators that creates the graph in Figure 6.

4.5.2 Graph Operators

The graph operators outlined in Table 7 are all binary operators which take a graph object on the left-hand and right-hand sides of the operator.

Operator	Explanation
graph + *graph* *graph* + *node*	Returns a graph that contains all of the nodes in the left-hand and right-hand graph or node
graph - *graph* *graph* - *node*	Removes the graph on the right-hand side of the operator from the graph on the left-hand side.
graph == *graph*	Returns whether the two graphs contain the same nodes.

Table 7: Graph Operators

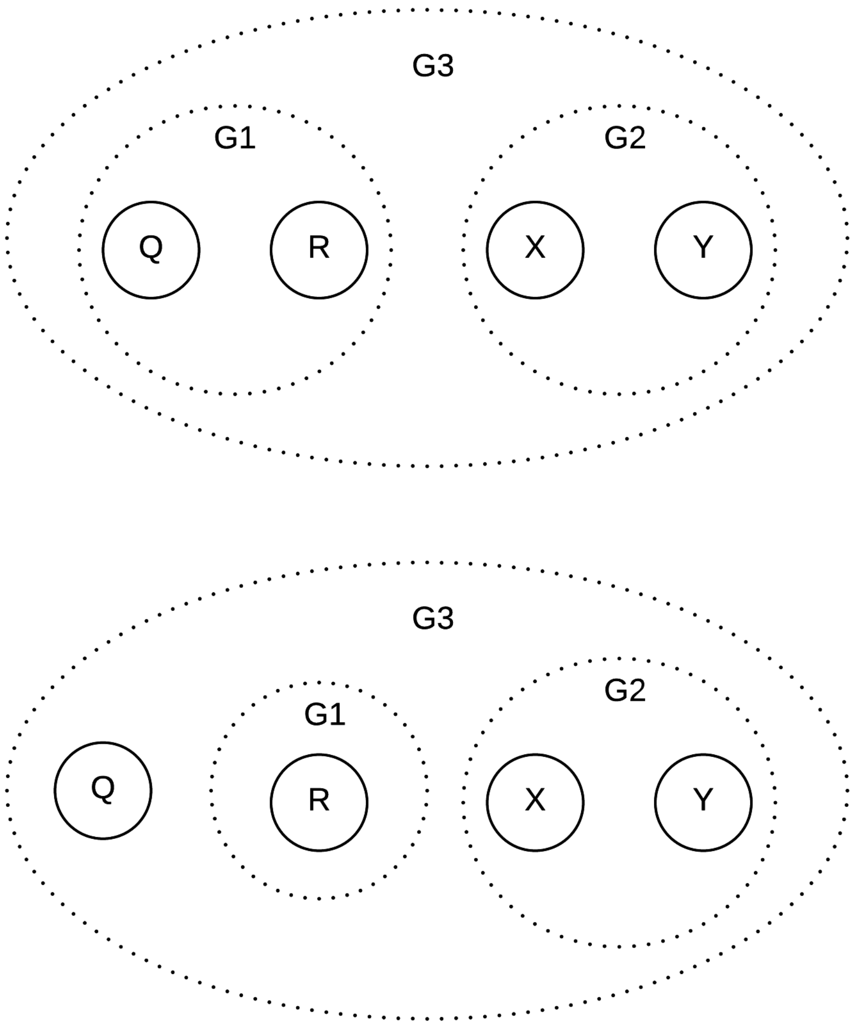


Figure 2: Example showing graphs and graph nesting. The bottom graph is the result of removing the node "Q" from the graph G1.

```
node X;
node Y;
node Q;
node R;

graph G1;
graph G2;
graph G3;

G2 = X + Y;
G1 = Q;
G3 = G1 + G2; # result is the top graph

G1 = G1 + R;
```

```
G1 = G1 - Q; # G1 is back to its original assignment
```

Listing 6: Shows the use of graph operators that creates the top graph in Figure 2 and then alters it to the bottom graph shown.

4.6 Built-in Functions

Along with these built in data types, d.o.t.s. provides some built in functions that serve to enhance the user's ability to easily script. These functions require no special declaration on the part of the user.

The `print` function is unusual in that unlike all functions defined by users, which must take a fixed number of arguments, `print` can take any variable number of arguments. It then writes a string representation of each of its arguments to standard out. The `print` function can directly operate on all the types in d.o.t.s. This means that even special types such as `dicts` and `lists` can directly be used as arguments to `print`.

The motivation for the `range` function was for users to be able to quickly iterate through values in orderly manner instead of having to worry about sometimes painful list iteration.

Syntax	Explanation
print(x, ...)	prints to standard output the string representation of a list of comma-separated values or variables.
range(*int_lower*, *int_upper*)	returns a list of the integers from *int_lower* to *int_upper*, inclusive. The first argument can be ommitted, in which case 0 will be used as the default value of *int_lower*. The data type of both arguments must be int.
len(*iterable_var*)	returns the length of the iterable variable
max(*iterable_var*)	Uses the ">" operator for detrming values; if ">" is undefined for the object, returns `null`. For lists, returns the element with greatest value. For dicts, returns the key of (key,value) pair with the greatest value.
min(*iterable_var*)	Uses the "<" operator for detrming values; if "<" is undefined for the object, returns `null`. For lists, returns the element with lowest value. For dicts, returns the key of (key,value) pair with the lowest value.

Table 8: Built-in Functions

```
list<int> x;
x = range(1, 3); # x = [1, 2, 3]
dict<string, num> y;
y["foo"] = 7;
y["bar"] = 8.9;

print("x: ", x, "\ny:", y);
/* prints out -->
  x: [1, 2, 3]
  y: ["foo": 7.000, "bar": 8.900]
```

```
*/

list<string> q = ["foo", "fah"];
len(q); # returns 2

dict<string, num> cityVals = {"chicago" : 2, "seattle" : 1, "nyc" : -8};
max(cityVals); # returns "chicago"
min(cityVals); # returns "nyc"
```

Listing 7: Shows the use of built-in functions.

4.7 Control Flow

As Section 5.4 includes example usage for each of the different types of control statements, this section omits a separate demonstration of their use.

4.7.1 Logical Expressions

Two of the types of control flow mechanisms, `if` statements and `while` loops, use logical expressions in their condition statement. In d.o.t.s., a logical expression consists of a sequence of expressions connected by a logical operator. The list of logical operators is given in Table 9.[6]

>
<
<=
>=
==
!=
&&
||

Table 9: Logical Operators

4.7.2 If Statements

```
/*
 condition -- a logical expression
*/
if (condition) {
 /* 1st code block */
}

if (condition) {

}
else {
 /* 2nd code block */
}
```

[6] *Note*: while the parser correctly deals with `&&` and `||`, both operators were not fully implemented in the translation side.

The condition of an `if` statement must be a logical expression. If the logical expression evaluates to true, the first code is executed. If the logical expression evaluates to false and there is an else block, then the 2nd code block is executed.

4.7.3 For Loops

```
/*
 iterable_var -- a variable with elements inside to iterate over
 temp_var -- variable with the type of elements in iterable_var
*/
for temp_var in iterable_var {
 /* code block */
}
```

A `for` loop may be executed on any iterable variable: i.e. lists, dicts, and graphs. The code within the for loop is executed once for each element inside the iterable variable. On each iteration, the next variable inside the iterable variable is assigned to the temporary variable.

The temporary variable in the for loop header is automatically created by the d.o.t.s. compiler, and must have a name not already used by a declared variable.it'

4.7.4 While

```
/*
 condition -- a logical expression
*/
while (condition) {
 /* code block */
}
```

The condition of a `while` loop must be a logical expression. On each iteration, if the condition evaluates to true, the code block is executed. If the condition evaluates to false, the while loop stops iterating and the program moves on to the next instructions. After the code is executed, another iteration is started.

4.8 User-defined Functions

```
def return_type function_name (arg1_datatype arg1_name, ...) {
 /* code */
}
```

In d.o.t.s., users are allowed to define their own functions, and user them in their code as they would a built-in function.

Elements in a function definition header:

- def – the keyword "def" must precede any function definition
- *return_ type* – the type of whatever value is returned by the function

  Return types are the same keywords and structure as the types used to declare variables.
- *function_ name* – the name used to call the function

  Function names have the same rules as variable names.
- parameter list – The parameter list is the comment-separated list of arguments inside the parentheses of a function definition header.

  Each argument in the parameter list must be of the form: *`datatype argname`*.

datatype is the type of the argument (ex. `num`). Data types are of the same structure as those used in variable declarations.

argname is the alias for the argument that is passed in to the function. The corresponding argument can be referenced using its *argname*. Argument names have the same restrictions as variable names.

An empty parameter list is used to define a function that takes 0 arguments.

```
def num foo (num x, string s) {
   print(s, ": ", x);
}

def string bar () {
   print("Hello there");
}

num y = 5;
string phrase = "Items to count: ";

foo(y, phrase); # prints --> Items to count: 5
bar(); # prints --> Hello there
```

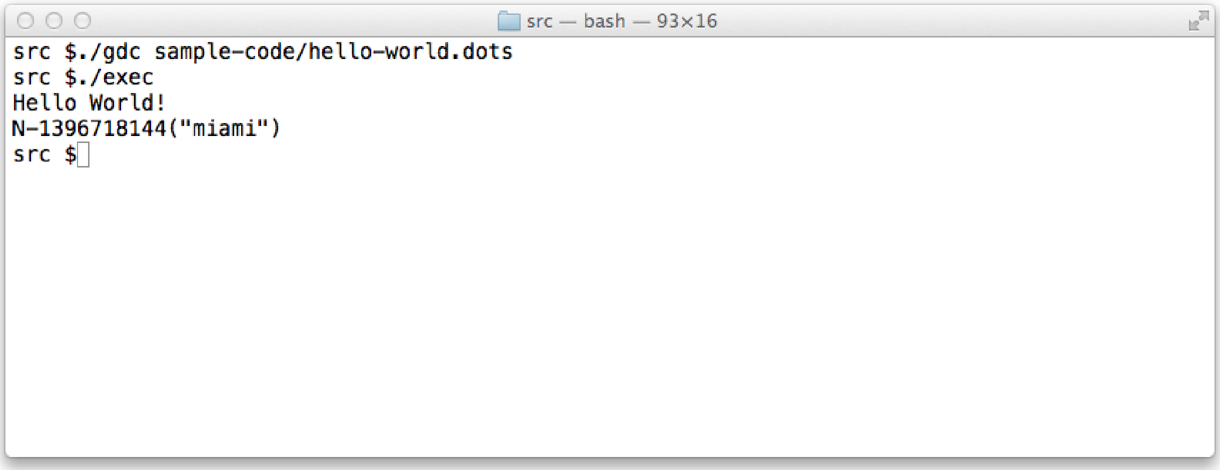
Listing 8: User-defined functions example

5 Example Code

5.1 Hello World

```
node x("miami");
string s = "Hello World!";
print(s, "\n", x, "\n");
```

Listing 9: Hello World program.



```
src $./gdc sample-code/hello-world.dots
src $./exec
Hello World!
N-1396718144("miami")
src $
```

Figure 3: Output after running hello-world.dots.

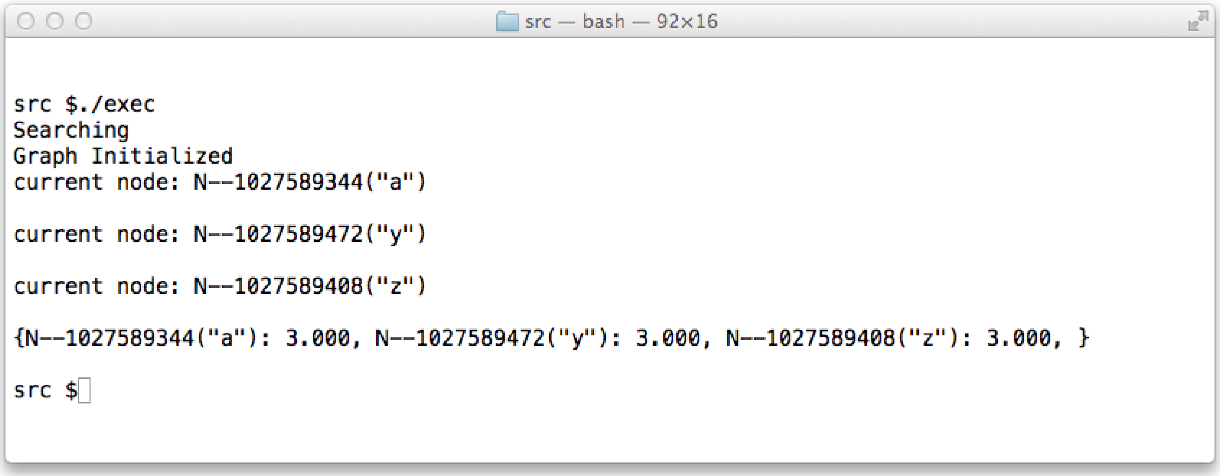
See Appendix E.1 for the compiled C code.

5.2 Breadth-First Search

In this section, we demonstrate a small working example of how you'd use d.o.t.s. to do graph algorithms.

```
#breadth first search

print("Searching\n");

def bool has_node (list<node> l, node x) {
  for (n in l) {
     if (n == x) {
       return true;
     }
  }
  return false;
}

graph g;

node x("x");
node y("y");
node z("z");

node a("a");
node b("b");
node c("c");

g = x + y;
g = g + z;

x -->[2] y;
x -->[1.5] z;
z -->[4] y;
y -->[2] c;
z -->[2.5] b;
c -->[.5] b;
x -->[333] a;
z -->[15] a;

print("Graph Initialized\n");

list<node> queue;
list<node> seen;
dict<node, num> dist;


node cur_node;
cur_node = x;

num curr_dist;
curr_dist = 0;

for (n in cur_node.oute()){
    print("current node: ", n, "\n", "\n");

    #print("Number of outgoing edges: ", len(cur_node.oute()));

    curr_dist = curr_dist + 1;

  if (has_node(seen, n) == false) {
     seen.enqueue(x);
     queue.enqueue(n);
     dist[n] = curr_dist;
  }
  cur_node = queue.peek();
  queue.dequeue();
}

print(dist, "\n\n");
```

Listing 10: Breadth-First Search Algorithm



```
src $./exec
Searching
Graph Initialized
current node: N--1027589344("a")

current node: N--1027589472("y")

current node: N--1027589408("z")

{N--1027589344("a"): 3.000, N--1027589472("y"): 3.000, N--1027589408("z"): 3.000, }

src $
```

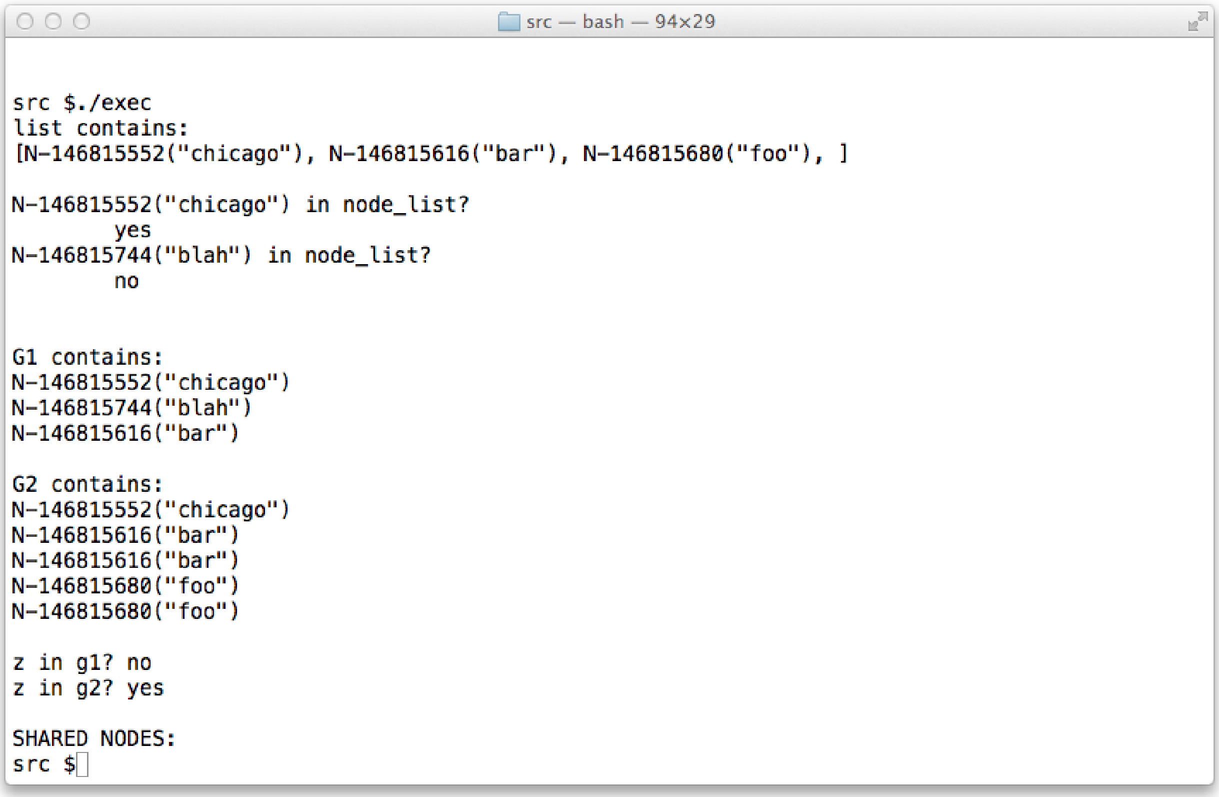
Figure 4: Output after running bst.dots.

See Appendix E.3 for the compiled C code.

5.3 User-Defined Functions Example

```
def string node_in_list (list<node> l, node x) {
  for (n in l) {
     if (n == x) {
        return "yes";
     }
  }
  return "no";
}

def string node_in_graph (graph g, node x) {
  for (n in g) {
    if (n == x) {
      return "yes";
    }
  }
  return "no";
}

def list<node> find_intersection(graph gg1, graph gg2) {
  list<node> result;

  for (n1 in gg1) {
    for (n2 in gg2) {
      if (n1 == n2) {
        result.enqueue(n1);
      }
    }
  }
  return result;
}

node x("chicago");
node y("bar");
node z("foo");
node w("blah");

/* NodeFunctions: */

list<node> node_list = [x, y, z];
string result;
print("list contains: \n", node_list, "\n", "\n");

result = node_in_list (node_list, x);
print(x, " in node_list?\n", "\t", result, "\n");

result = node_in_list (node_list, w);
print(w, " in node_list?\n", "\t", result, "\n");

print("\n\n");

/* regular graph declaration */
graph g1;
g1 = x + w;
g1 = g1 + y;

print("G1 contains:\n");
for (n1 in g1) {
  print(n1, "\n");
}
print("\n");

/* fancy graph declaration */
graph g2 = {
  x -- y,
  y --> z,
 z -->[22.3] x
};

print("G2 contains: \n");
for (n2 in g2) {
 print(n2, "\n");
}
print("\n");

result = node_in_graph(g1, z);
print("z in g1? ", result, "\n");

result = node_in_graph(g2, z);
print("z in g2? ", result, "\n");

/* graph function */
list<node> union;
union = find_intersection(g1, g2);

print("\nSHARED NODES:\n");
```

Listing 11: Simple example showing off different features of the language including userdefined functions.



```
src $./exec
list contains:
[N-146815552("chicago"), N-146815616("bar"), N-146815680("foo"), ]

N-146815552("chicago") in node_list?
        yes
N-146815744("blah") in node_list?
        no


G1 contains:
N-146815552("chicago")
N-146815744("blah")
N-146815616("bar")

G2 contains:
N-146815552("chicago")
N-146815616("bar")
N-146815616("bar")
N-146815680("foo")
N-146815680("foo")

z in g1? no
z in g2? yes

SHARED NODES:
src $
```

Figure 5: Output after running sample01.dots.

See Appendix E.2 for the compiled C code.

5.4 Dijkstra's Algorithm

The Dijkstra's Algorithm example in Listing 12 was our end-goal for the d.o.t.s. compiler. Unfortunately, although pieces of the necessary elements have been implemented and are working, we didn't have enough time to implement all the pieces needed for this example to run.

```
/*
 * Djikstra's algorithm: calculates shortest paths starting
 * from the source node and returns a dict of the lowest cost
 * to destination nodes
 */

def dict<node, num> relax (node u, dict<node, num> w) {
      for (v in u.oute()) {
         if (w[v] > u.oute()[v]) {
            w[v] = u.oute()[v];
         }
      }
      return w;
}

def dict<node, num> dijkstra (graph G, node source){
   dict<node, num> S;
   dict<node, num> Q;

   for (n in G) {
      Q[n] = INF;
   }
   Q[source] = 0;

   while (len(Q) != 0) {
      node u;
      u = min(Q);
      num w;
      w = Q[u];
      Q.remove(u);
      S[u] = w;
      Q = relax(u, Q);
   }
   return S;
}

/* Graph set-up */
node x("dc");
node y("chicago");
node z("philly");
node q("nyc");
node r("boston");

graph g1 = {
   x --[2] z,
   z --[2] q,
   q --[3] r,
   z --[9] r,
   x --[8] y,
   y --[9] r
};

/* end Graph set-up */

/* find the min costs from "philly" to all other cities: */
dict<node, num> min_costs;
min_costs = dijkstra(g1, z);
```

Listing 12: Dijkstra's Algorithm

Now that we have defined the `bfs` and `dijkstra` functions, we can use them to find shortest paths in an actual instance of a graph.

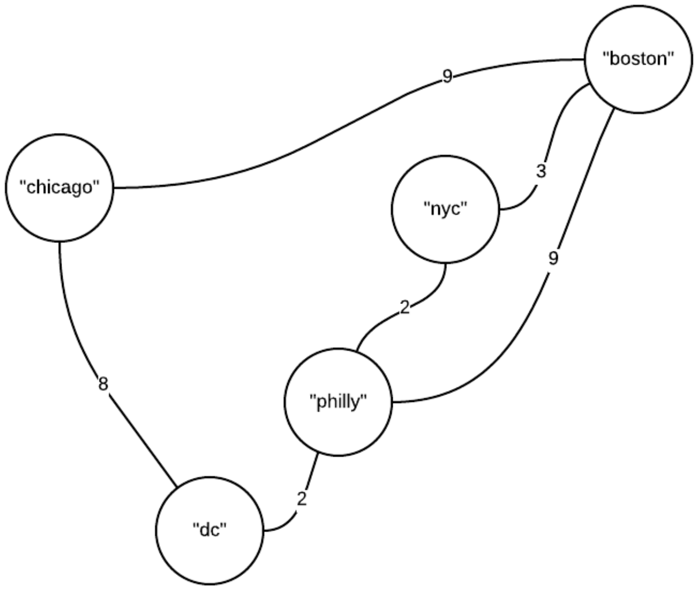


Figure 6: Visual representation of graph created in Listing 13.

```
/* Graph set-up */
node x("dc"), y("chicago"), z("philly"), q("nyc"), r("boston");

graph g1 = {
   x --[2] z;
   z --[2] q;
   q --[3] r;
   z --[9] r;
   x --[8] y;
   y --[9] r;
};
/* end Graph set-up */

# find the min costs from "philly" to all other cities:
```

```
dict<node, num> min_costs = dijkstra(g1, z);

# find path with minimum number of hops from "philly" to "chicago":
list<node> min_path = bfs(g1, z, y);
```

Listing 13: Using user-defined functions.

6 Project Development and Lifetime

6.1 Project Plan

Our team planned a weekly meeting usually on Sunday nights. During our in-person meeting time we would spend roughly ten minutes going over the checklist of things that needed to get done and then worked on those either individually or paired. We also kept each other up to date on work we'd done during the week with biweekly "standups" via Slack in which we updated each other on things we were working on, things that we needed to work on, and blockers. Each of those updates were posted to a respective Slack channel. For example, AST updates were posted on the AST channel, type converter issues to the Typeconverter channel, etc.

6.2 Planning and Synchronization

Once we had a better understanding of what we wanted to accomplish in terms of language features, we wrote an extensive Language Reference Manual including all of our goals, language features, and expected use cases. Aquila, our TA, met with us to go over the status of our LRM and gave us feedback as to which language to compile to and we ended up choosing C instead of C++ to avoid being considered a "java to java" language. From there we created a GitHub repository in order of us to handle version control and merge conflict issues.

6.3 Project Development

Initially we began by pushing all of our code to the `master` branch. However we quickly realized that this did not allow us to roll back changes, if necessary, to the last working version. Instead, we decided to create several branches but primarily the `clib` and `compile` branches which continuously got merged back into combined branches when they fit the following criteria:

1. dots source code compiles to `.dotc` file
2. dots source code compiles to `./exec` file (our C executable)
3. positive unit tests have been written for the particular language feature
4. negative unit tests have been written for the particular language feature
5. expected output file written to match the positive and negative tests
6. code passes when you run `runtest.py`

branches were only merged back to the feature branch when the entire file you were working on compile with no shift reduce, warnings, or error messages. This allowed us to test each new file we added incrementally so debugging was easier. Additionally, to allow testing run more smoothly we attempted to work on the `print` functions first for each stage of development in order to run the aforementioned test on each new language feature.

6.4 Development Environment

Dots Compiler Written Using:

- OCaml
- OCamllex
- OCamlyacc

Testing Performed Using:

- menhir
- Python 2.7

  dependencies: subprocess32

C Libary Written Using:

- C (on unix)
- gcc

Final Compilation Done Using:

- gcc

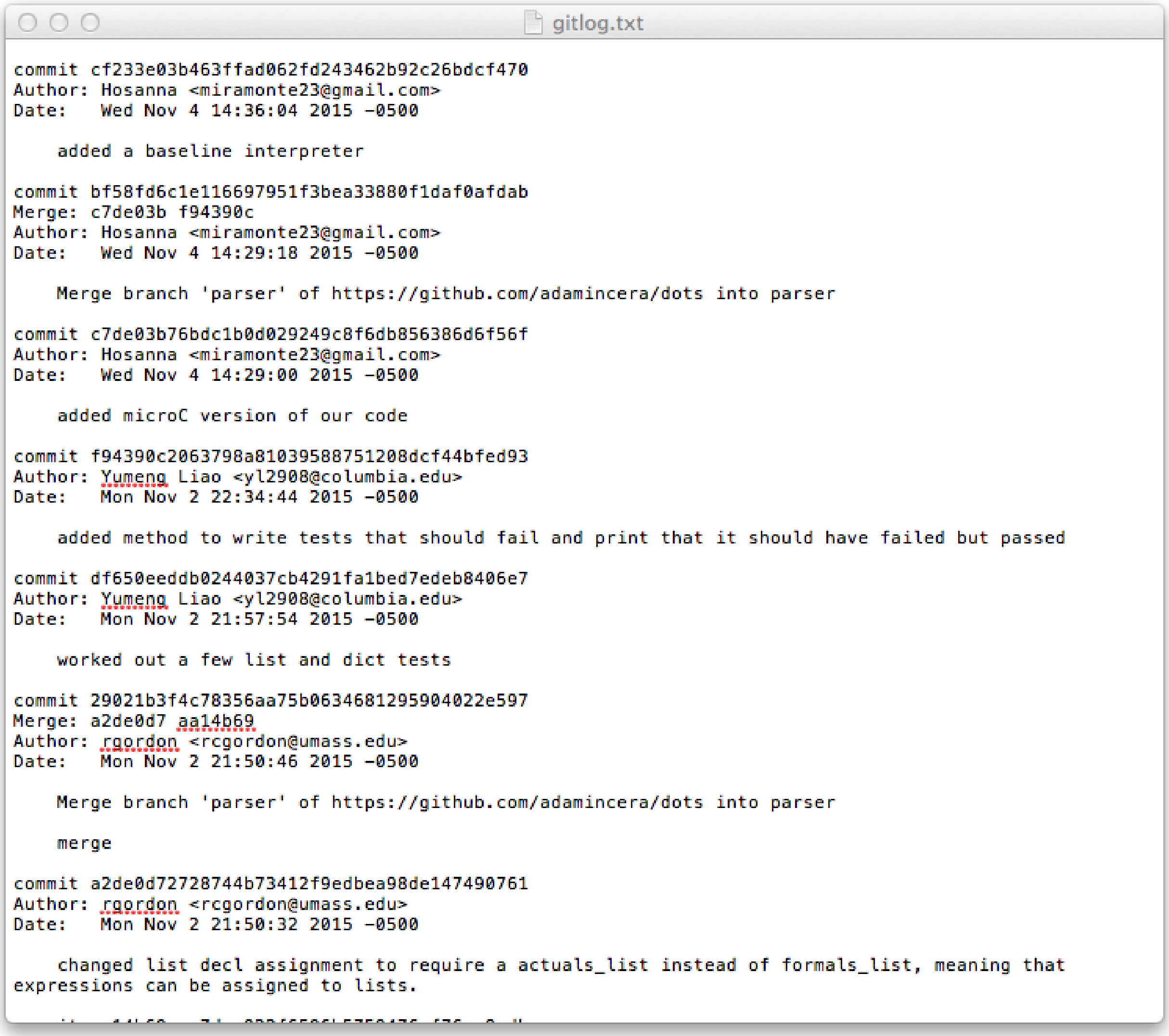
Note: On Mac (which was our development environment), `gcc` is really an alias for `clang`. So everywhere we mention `gcc`, behind the scenes it's actually using `clang`.

6.5 Style Guide

1. Comment particularly esoteric sections of code
2. Indent code hierarchically
3. For long parameter lists, put each parameter on a separate line
4. For long lines of code, put sub-pieces on a separate line

6.6 Project Log

Figure 7 shows an excerpt from our git commit history. For the full git log, see Appendix F.



```
commit cf233e03b463ffad062fd243462b92c26bdcf470
Author: Hosanna <miramonte23@gmail.com>
Date:   Wed Nov 4 14:36:04 2015 -0500

    added a baseline interpreter

commit bf58fd6c1e116697951f3bea33880f1daf0afdab
Merge: c7de03b f94390c
Author: Hosanna <miramonte23@gmail.com>
Date:   Wed Nov 4 14:29:18 2015 -0500

    Merge branch 'parser' of https://github.com/adamincera/dots into parser

commit c7de03b76bdc1b0d029249c8f6db856386d6f56f
Author: Hosanna <miramonte23@gmail.com>
Date:   Wed Nov 4 14:29:00 2015 -0500

    added microC version of our code

commit f94390c2063798a81039588751208dcf44bfed93
Author: Yumeng Liao <yl2908@columbia.edu>
Date:   Mon Nov 2 22:34:44 2015 -0500

    added method to write tests that should fail and print that it should have failed but passed

commit df650eeddb0244037cb4291fa1bed7edeb8406e7
Author: Yumeng Liao <yl2908@columbia.edu>
Date:   Mon Nov 2 21:57:54 2015 -0500

    worked out a few list and dict tests

commit 29021b3f4c78356aa75b0634681295904022e597
Merge: a2de0d7 aa14b69
Author: rgordon <rcgordon@umass.edu>
Date:   Mon Nov 2 21:50:46 2015 -0500

    Merge branch 'parser' of https://github.com/adamincera/dots into parser

    merge

commit a2de0d72728744b73412f9edbea98de147490761
Author: rgordon <rcgordon@umass.edu>
Date:   Mon Nov 2 21:50:32 2015 -0500

    changed list decl assignment to require a actuals_list instead of formals_list, meaning that
expressions can be assigned to lists.
```

Figure 7: Excerpt from our git commit history.

6.7 Roles and Responsibilities

Hosanna Fuller	Project Manager – In charge of scheduling, divying out tasks, keeping us on track, sprint goals. OCaml Developer – develop OCaml portion of the d.o.t.s. compiler
Adam Incera	C Guru – Sole author of C Libraries that underlie the d.o.t.s. compiler OCaml Assistant – Helped with development of the analyzer.ml file.
Yumeng Liao	Tester – In charge of writing the test suites and test cases. Debugger of OCaml Compiler. OCaml Developer – develop OCaml portion of the d.o.t.s. compiler
Rachel Gordon	Dots Guru – In charge of d.o.t.s. language and feature design. OCaml Guru – Develop and design OCaml Code, assist team members with OCaml development and understanding

Table 10: Roles and Responsibilities

7 Compiler Design

Conceptually, the d.o.t.s. compiler is split into four pieces:

1. Input Processing – Interpreting the `.dots` source file into a basic abstract syntax tree.

   Files: scanner.mll, parser.mly, ast.ml

2. DOTS Handling – Processing the DOTS syntax trees, including semantic checking.

   Files: ast.ml, Sast.ml, typeConverter.ml

3. C Handling – Converting DOTS abstract syntax trees into C abstract syntax trees, and converting that tree into actual C code.

   Files: Sast.ml, analyzer.ml, translate.ml

4. Binary Compilation – turning the C code into a binary executable.

   gcc

Figure 8 gives a high-level look at the compiler, while Sections 7.1 - 7.3 explain the first three stages in more depth. The final stage, Binary Compilation, consists of running `gcc` with the C Library on the compiled C code.

Hosanna Fuller, Yumeng Liao, and Rachel Gordon worked on all of the OCaml files. Adam Incera also worked on analzyer.ml. The C libraries were done entirely by Adam Incera; accordingly, he worked less with the OCaml files.

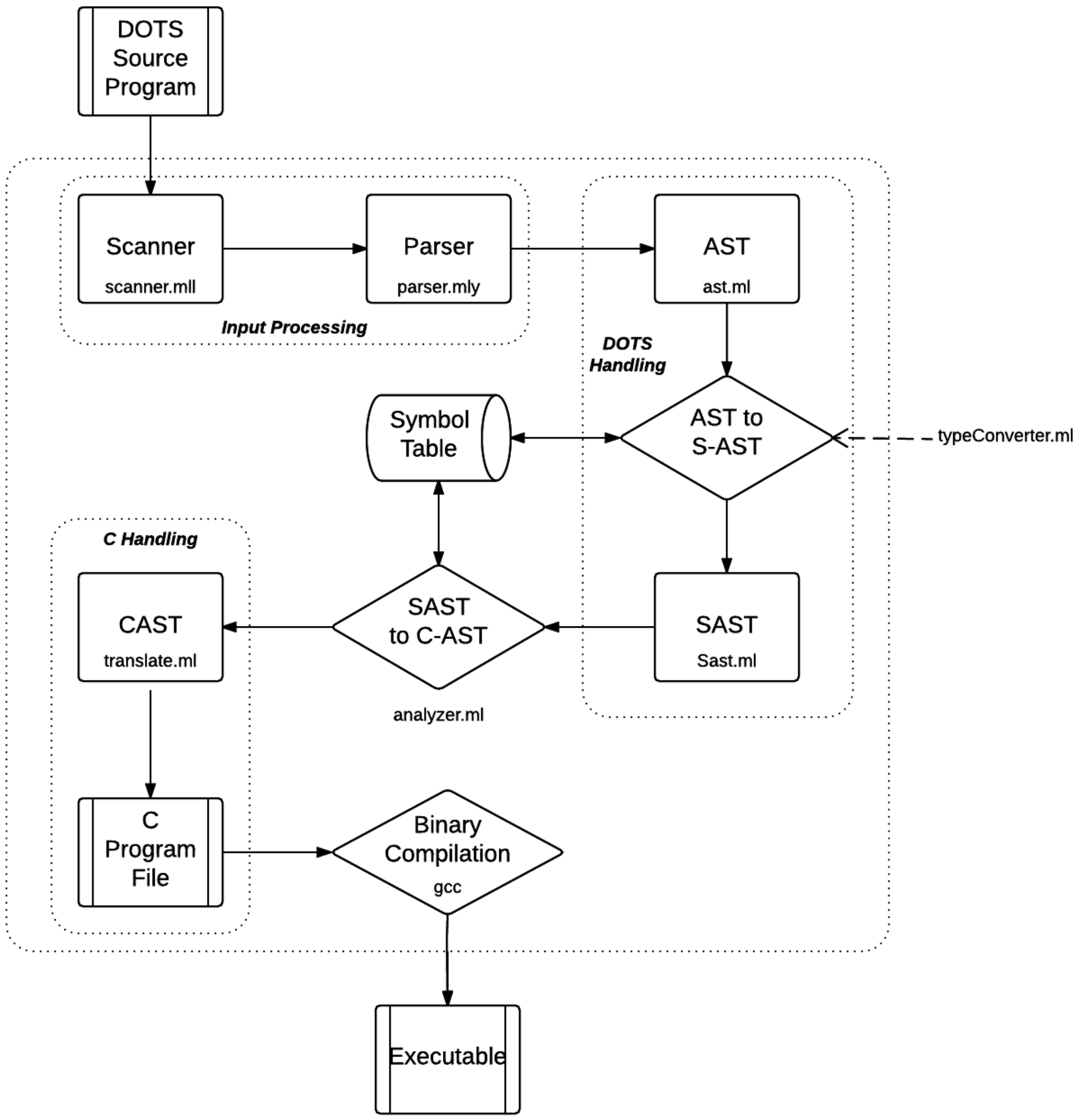


Figure 8: Visual representation of the d.o.t.s. compiler.

7.1 Input Processing

This stage consists of converting a source file written in DOTS into a series of tokens that can be easier processed by the compiler.

The scanner makes sure that all tokens in the input program are valid, while the parser checks on a basic level whether tokens have been used in an appropriate ordering.

7.2 DOTS Handling

This stage consists of semantically checking the basic AST and converting the fully-semantically checked into an SAST (aka a semantically typed AST).

The files `ast.ml` and `Sast.ml` define the elements in a basic AST and semantically typed AST (SAST) respectively. The `typeConverter.ml` file contains the functions which recursively walk an AST program and convert it into an SAST program. During the AST walk, the converter also semantically checks the d.o.t.s. program.

After parsing and scanning have finished, the d.o.t.s. compiler has a basic Abstract Syntax Tree. This basic AST is not ideal, however, for either translating into C or semantically checking the d.o.t.s. code, as the parser cannot fully type each element of the AST. This means that the elements in the AST also are not fully typed. Instead, we needed to convert the AST into a form of the DOTS syntax tree that did associate each element with its type.

The recursive definition of elements in an AST (expr, stmt, program) makes this type checking easier. For elements containing or referencing other objects, the converter can first semantically check and type the AST subelements into an SAST element and then easily extract the appropriate data type to check the parent element.

Using the types, the converter can that check elements of certain types are used in the appropriate places. For example, if the converter is checking an `assign` statement, it will check that the type of the object being assigned matches what is being assigned to it.

This stage also makes sure that all referenced variables have been declared and that the same variable name is not overwritten by a subsequent declaration. To keep track of what variables have already been assigned and which types they are, the converter keeps track of several symbol tables in an environment variable. The first table maps variable names to types, the second maps variable names to indices, the third maps function names to types and the fourth maps function names to indices.

The two index tables are used in the C translation. This is because a d.o.t.s. program has few restrictions on variable names in terms of reserved words. Because of this, it is possible that a user might declare variables and functions which conflict with names in the underlying C libraries. To avoid this problem, we wanted to use generic variable and function names in our converted C file. This way, each function and variable name has a unique number identifying it and can be referenced simply as `v1, f1,` etc.

Once the semantic checking and conversion to SAST is finished, the compiler is guaranteed that the input DOTS program is valid.

7.3 C Handling

This stage consists of converting the SAST into an equivalent C AST representation of a C program, and converting that C AST into actual C code.

Because the structure of a d.o.t.s. program differs from that of a C program, the SAST required some special handling before it could be converted into a C AST. The main issue was that in d.o.t.s., users can execute any number of statements outisde functions, in the manner of a scripting language. C, however, only allows global variable declarations outside of functions. Most scripting-style commands can simply be put into the main function in C. And because order is preserved, and the main function is run first, the executing C program's order will also match. But if all of the script-style commands were dumped into the main function in C, then variables which were treated like globals in d.o.t.s. would become local in the C program, making references to them in user-defined functions fail.

To solve this issue, we first walked through a completed SAST and separated the program into three pieces: variable declarations outside of functions, function declarations and their bodies, and all other statements. The compiler then runs the C conversion on each of the three pieces sequentially. Declaring all globally in the C program that were declared outside of functions in the d.o.t.s. program allows the converted C functions to use them.

After splitting the SAST program into those three programs, the compiler can then convert it into a C AST. The `analyzer.ml` file walks down the SAST and outputs corresponding elements of a C AST. The end result is full C AST program object. The `translate.ml` file can then walk through its C AST and essentially pretty-print a C program.

8 C Library

In order to make it feasible to generate C code that creates and manipulates graphs and other complex data structures, we wrote a graph API in C from which the compiler can generate function calls. The library contains implementations for data types for graphs, nodes, lists, and dictionaries, as well as all the standard operations associated with those data structures.

One of the most difficult portions to implement in C was the `dict` data type. The solution we chose is a simplified version of Python's underlying implementation of its ditionaries, where the dictionary is essentially an array of linked lists of key-value pairs. In order to accomplish hashing, we implemented simple datatype-specific hash algorithms.

The `get()` function followed these steps:

- Hash the key using the appropriate hash function to obtain the hashed key k.
- Iterate through the list beginning at the kth element in the array, checking the key of each key-value pair against the unhashed key.
- If a match is found, return the correponding value.
- If the end of the list is reached, return NULL.

The `put()` function followed these steps:

- Hash the key using the appropriate hash function to obtain the hashed key k.
- Iterate through the list beginning at the kth element in the array, checking the key of each key-value pair against the unhashed key.
- If a match is found, copy the value into the value field of the entry.
- If the end of the list is reached, insert a new node containing the key-value pair.

The following hashing algorithms were used (`TABLE_SIZE` is a constant defining the size of the array inside the dictionary):

- `string`: sum the characters and reduce mod `TABLE_SIZE`.
- `num`: reduce mod `TABLE_SIZE`.
- node: bit shift the memory address of the node three to the right (because pointers are eight-byte-aligned, so the rightmost bits were always 0), then reduce mod `TABLE_SIZE`. The memory address was used because node equality in d.o.t.s.is defined as physical equality.
- `graph`: sum the memory addresses of the nodes it contains, bit shift three to the right, and reduce mod `TABLE_SIZE`.
- `other`: anything else could be hashed by reducing its memory address mod `TABLE_SIZE`.

The other major difficulty encountered was emulating d.o.t.s.' generics (`dict` and `list`) in C. Because C doesn't have built-in templates, we used the next best thing, which was `void *`s and function pointers. Each `list` and `dict` operation was written as a generalized static function that was datatype-agnostic and took in a function pointer to perform comparison, copying, or whatever type-specific operation was required.

This function was then accessed through type-specific wrapper functions that cast the datatype as a `void *` and passed in the appropriate function pointer.

9 Testing

9.1 Test Suites

To help with writing the parser, we wrote tests in .txt files that consisted of a string of tokens as defined by our Ocamllex parser file. These were piped into our Menhir testing script to ensure that while adding new rules to the parser nothing that was working would be broken again.

Shortcut to Menhir testing script: D.2

After we started to generate C code, we wrote unit tests in the form of small individual snippets of d.o.t.s. code to test that a particular feature was working. Such features included: operators, function and variable declarations, looping, etc. These snippets were organized in sub-directories according to what core functionality they were testing. Test scripts in the "complex" directory contain programs that are more practical and combine multiple features.

Shortcut to main testing script: D.1

Note: to run tests, make sure to use Python's pip package manager to install the required packages, using "pip install -r requirements.txt" from the src directory.

9.2 Test Automation

Both testing scripts made use of Python's subprocess module to open the appropriate processes. The Menhir script runs all tests in the "menhir-tests" directory, reading each line of each .txt file so as to see which tests are supposed to pass and which are supposed to fail. The test script ignores lines beginning with "***" and looks for failure or a pass condition on lines beginning with "f**". Different flags in running the test script provide different forms of output such as: suppressing errors, printing the full tree, etc.

The main testing script looks through the `dtest` directory, which is dedicated to tests that should pass (aka positive tests), and the `ntest` directory, which is dedicated to tests that should fail (aka negative tests). For both, it walks through all subdirectories and builds a .c file, executable, and processes the gdc (the shell script that ties together all parts of our compiler) output. It then runs the executable (if there is one) and compares the output to a pre-written `.out` file. A difference between expected and actual output means failure. Such differences are recorded in `.dif` files. Specifying certain flags will clean up intermediate files and will only print the full list of results. It's as simple as running "python runtest.py".

9.3 Source to Target

Source program (bst.dots):

```
#breadth first search

print("Searching\n");

def bool has_node (list<node> l, node x) {
  for (n in l) {
     if (n == x) {
       return true;
     }
  }
  return false;
}

graph g;

node x("x");
node y("y");
node z("z");

node a("a");
node b("b");
node c("c");

g = x + y;
g = g + z;

x -->[2] y;
x -->[1.5] z;
z -->[4] y;
y -->[2] c;
z -->[2.5] b;
c -->[.5] b;
x -->[333] a;
z -->[15] a;

print("Graph Initialized\n");

list<node> queue;
list<node> seen;
dict<node, num> dist;


node cur_node;
cur_node = x;

num curr_dist;
curr_dist = 0;

for (n in cur_node.oute()){
   print("current node: ", n, "\n", "\n");

   #print("Number of outgoing edges: ", len(cur_node.oute()));

   curr_dist = curr_dist + 1;

  if (has_node(seen, n) == false) {
     seen.enqueue(x);
     queue.enqueue(n);
     dist[n] = curr_dist;
  }
  cur_node = queue.peek();
  queue.dequeue();
}
```

```c

print(dist, "\n\n");
```

Listing 14: Breadth-First Search Algorithm

Target program: bst.dots.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dict.h>
graph_t * v1 = NULL;
node_t * v2;
node_t * v3;
node_t * v4;
node_t * v5;
node_t * v6;
node_t * v7;
list_t * v8 = NULL;
list_t * v9 = NULL;
entry_t** v10 = NULL;
node_t * v11;
float v12;
int f6 (list_t * v111, node_t * v112)
{
list_t ** v113;
v113 = &(v111);
node_t * v115;
list_t * v114 = NULL;
for (v114 = *(v113); v114; v114 = (v114)->next) {
v115 = *((node_t **)((v114)->data));
node_t ** v116;
v116 = &(v115);
node_t ** v117;
v117 = &(v112);
if (*(v116) == *(v117)) {
int* v118;
v118 = malloc(sizeof(int));
*(v118) = 1;
return *(v118);
} else {

}
}
int* v119;
v119 = malloc(sizeof(int));
*(v119) = 0;
return *(v119);
}

int main (int argc, char ** argv)
{
char ** v13;
```

```
v13 = malloc(sizeof(char *));
*(v13) = malloc(strlen("Searching\n") + 1);
strcpy(*(v13), "Searching\n");
printf("%s", *(v13));;
char ** v14;
v14 = malloc(sizeof(char *));
*(v14) = malloc(strlen("x") + 1);
strcpy(*(v14), "x");
v2 = init_node("");
(v2)->data = *(v14);
char ** v15;
v15 = malloc(sizeof(char *));
*(v15) = malloc(strlen("y") + 1);
strcpy(*(v15), "y");
v3 = init_node("");
(v3)->data = *(v15);
char ** v16;
v16 = malloc(sizeof(char *));
*(v16) = malloc(strlen("z") + 1);
strcpy(*(v16), "z");
v4 = init_node("");
(v4)->data = *(v16);
char ** v17;
v17 = malloc(sizeof(char *));
*(v17) = malloc(strlen("a") + 1);
strcpy(*(v17), "a");
v5 = init_node("");
(v5)->data = *(v17);
char ** v18;
v18 = malloc(sizeof(char *));
*(v18) = malloc(strlen("b") + 1);
strcpy(*(v18), "b");
v6 = init_node("");
(v6)->data = *(v18);
char ** v19;
v19 = malloc(sizeof(char *));
*(v19) = malloc(strlen("c") + 1);
strcpy(*(v19), "c");
v7 = init_node("");
(v7)->data = *(v19);
node_t ** v20;
v20 = &(v2);
node_t ** v21;
v21 = &(v3);
graph_t ** v22;
v22 = malloc(sizeof(graph_t *));
*(v22) = (node_plus_node(*(v20), *(v21)));
graph_t ** v23;
v23 = &(v1);
*(v23) = *(v22);
graph_t ** v24;
v24 = &(v1);
node_t ** v25;
v25 = &(v4);
```

```c
graph_t ** v26;
v26 = malloc(sizeof(graph_t *));
*(v26) = (graph_plus_node(*(v24), *(v25)));
graph_t ** v27;
v27 = &(v1);
*(v27) = *(v26);
float* v28;
v28 = malloc(sizeof(float));
*(v28) = 2;
connect_dir_weighted (v2, v3, *(v28));
float* v29;
v29 = malloc(sizeof(float));
*(v29) = 1.5;
connect_dir_weighted (v2, v4, *(v29));
float* v30;
v30 = malloc(sizeof(float));
*(v30) = 4;
connect_dir_weighted (v4, v3, *(v30));
float* v31;
v31 = malloc(sizeof(float));
*(v31) = 2;
connect_dir_weighted (v3, v7, *(v31));
float* v32;
v32 = malloc(sizeof(float));
*(v32) = 2.5;
connect_dir_weighted (v4, v6, *(v32));
float* v33;
v33 = malloc(sizeof(float));
*(v33) = .5;
connect_dir_weighted (v7, v6, *(v33));
float* v34;
v34 = malloc(sizeof(float));
*(v34) = 333;
connect_dir_weighted (v2, v5, *(v34));
float* v35;
v35 = malloc(sizeof(float));
*(v35) = 15;
connect_dir_weighted (v4, v5, *(v35));
char ** v36;
v36 = malloc(sizeof(char *));
*(v36) = malloc(strlen("Graph Initialized\n") + 1);
strcpy(*(v36), "Graph Initialized\n");
printf("%s", *(v36));;

v11 = init_node("");
(v11)->data = "";
node_t ** v37;
v37 = &(v2);
node_t ** v38;
v38 = &(v11);
*(v38) = *(v37);
float* v39;
v39 = malloc(sizeof(float));
*(v39) = 0;
```

```c
float* v40;
v40 = &(v12);
*(v40) = *(v39);
node_t ** v41;
v41 = &(v11);
entry_t*** v42;
v42 = &((*(v41))->out);
int v69;
entry_t* v43;
void* v44;
if (*(v42)) {
for (v69 = 0; v69 < TABLE_SIZE; v69 = v69 + 1) {
for (v43 = (*(v42))[v69]; v43; v43 = (v43)->next) {
v44 = (v43)->key;
char ** v70;
v70 = malloc(sizeof(char *));
*(v70) = malloc(strlen("current node: ") + 1);
strcpy(*(v70), "current node: ");
printf("%s", *(v70));
node_t ** v71;
v71 = &(v44);
printf("%s", "N-");
printf("%d", (int)(*(v71)));
printf("%s", "(\"");
printf("%s", (char *)((*(v71))->data));
printf("\")");
char ** v72;
v72 = malloc(sizeof(char *));
*(v72) = malloc(strlen("\n") + 1);
strcpy(*(v72), "\n");
printf("%s", *(v72));
char ** v73;
v73 = malloc(sizeof(char *));
*(v73) = malloc(strlen("\n") + 1);
strcpy(*(v73), "\n");
printf("%s", *(v73));;
float* v74;
v74 = &(v12);
float* v75;
v75 = malloc(sizeof(float));
*(v75) = 1;
float* v76;
v76 = malloc(sizeof(float));
*(v76) = (*(v74) + *(v75));
float* v77;
v77 = &(v12);
*(v77) = *(v76);

list_t ** v78;
v78 = &(v9);
node_t ** v79;
v79 = &(v2);
list_t ** v80;
v80 = malloc(sizeof(void));
```

```
*(v78) = node_add_back(*(v78), *(v79));;
list_t ** v81;
v81 = &(v8);
node_t ** v82;
v82 = &(v44);
list_t ** v83;
v83 = malloc(sizeof(void));
*(v81) = node_add_back(*(v81), *(v82));;
entry_t*** v84;
v84 = &(v10);
node_t ** v85;
v85 = &(v44);
float* v86;
v86 = &(v12);
node_t * v88;
v88 = *(v85);
float v87;
v87 = *(v86);
*(v84) = put_node(*(v84), (node_t *)(v88), (void*)(&(v87)));
list_t ** v89;
v89 = &(v8);
node_t ** v90;
v90 = peek(*(v89));
node_t ** v91;
v91 = &(v11);
*(v91) = *(v90);
list_t ** v92;
v92 = &(v8);
void* v93;
*(v92) = pop(*(v92));;
}
}
} else {

}
printf("{");
entry_t*** v95;
v95 = &(v10);
int v104;
entry_t* v96;
void* v97;
if (*(v95)) {
for (v104 = 0; v104 < TABLE_SIZE; v104 = v104 + 1) {
for (v96 = (*(v95))[v104]; v96; v96 = (v96)->next) {
v97 = (v96)->key;
node_t ** v105;
v105 = &(v97);
printf("%s", "N-");
printf("%d", (int)(*(v105)));
printf("%s", "(\"");
printf("%s", (char *)((*(v105))->data));
printf("\")");;
char ** v106;
v106 = malloc(sizeof(char *));
*(v106) = malloc(strlen(": ") + 1);
strcpy(*(v106), ": ");
printf("%s", *(v106));;
entry_t*** v107;
v107 = &(v10);
node_t ** v108;
v108 = &(v97);
float* v109;
v109 = (float*)(get_node(*(v107), *(v108)));
printf("%.3f", *(v109));;
char ** v110;
v110 = malloc(sizeof(char *));
*(v110) = malloc(strlen(", ") + 1);
strcpy(*(v110), ", ");
printf("%s", *(v110));;
}
}
} else {

}
printf("}");
char ** v111;
v111 = malloc(sizeof(char *));
*(v111) = malloc(strlen("\n\n") + 1);
strcpy(*(v111), "\n\n");
printf("%s", *(v111));;
}
```

Listing 15: bst.dots.c

10 Lessons Learned

10.1 Hosanna's Advice

The sooner you dig into OCaml, the better off you are going to be. Even if you have no idea how to write the compiler or some other assignment you have. Start building small programs because they will most likely come in handy some time in the future.

Similarly, try to truncate problems out into smaller sub-problems and try to get those parts to compile. It's easy to be ambitious in the beginning and try to get your entire parser working in one fell swoop, but its better to test incrementally. Especially with OCaml's error messages, you need to be careful and build your code in a modular manner.

10.2 Adam's Advice

Keep up with what the rest of your team is doing! I spent a large part of the semester implementing the C library, and by the time I finished it, the code base was big enough that I had to spend a bunch more time just getting up to speed on what everyone else had written.

Do the calculator assignment. Get comfortable with Ocaml as soon as you possibly can, so that when the time comes you can focus on actually writing your compiler instead of trying to decipher a language you don't understand.

10.3 Rachel's Advice

If you only ever do one homework assignment, whatever you do make it the calculator assignment. Doing that assignment teaches you the basics of how to think in OCaml, and if you don't do it, then the learning curve when you go to do your project is going to be huge.

Prioritize implementation of your language based on your end-goal example code. If your language design includes W, X, Y, Z, but the program you want to be able to demo only uses X and Y, start with X and Y, don't waste time implementing Z.

Sit down and design your compiler on paper before you start coding. Think about specific examples of what you want to be able to write in your language and what that's going to mean in terms of what information you need encoded in your ASTs. If you only design it as you write it, you're going to end up having to rewrite portions of the compiler multiple times.

Don't develop your underlying libraries and compiler in isolation. They're interdependent, so if you don't keep that in mind when you're designing your system, the different components won't work well together and might not be able to express what you need them to.

10.4 Yumeng's Advice

Don't try to stay within the strict definitions of the role that you are assigned within the team, because each role must know everything about what everyone else is doing in order to put out quality work. As the tester, early on I fell into the trap of writing tests that weren't immediately relevant to what the rest of the team was doing because I didn't understand how the compiler was working. As a result, my first few tests weren't useful to the team. I was only able to write quality tests only after I knew the intricacies of what everyone else was doing and after I'd put work into the compiler.

Try to find people you like, and more importantly, respect tremendously. It'll be much easier to treat each other with kindness, and trust that your teammates will pull their weight.

Appendices

A OCaml Compiler Files

A.1 scanner.mll

```
{ open Parser }

let num = ['0'-'9']+
let num_regex = '-'?(num*'.'num+) | (num+('.'num*)?)

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"  { multicomment lexbuf } (* Multi-Line Comments *)
| "#"   { comment lexbuf }    (* Single-Line Comments *)
| '('   { LPAREN }
| ')'   { RPAREN }
| '{'   { LBRACE }
| '}'   { RBRACE }
| '['   { LBRACKET }
| ']'    { RBRACKET }
| ';'    { SEMI }
| ':'    { COLON }
| ','    { COMMA }
| '+'    { PLUS }
| '-'    { MINUS }
| '*'    { TIMES }
| '/'    { DIVIDE }
| '='    { ASSIGN }
| '.'    { DOT }
| "&&"   { LOGAND }
| "||"   { LOGOR }
| "--"   { UEDGE }
| "-->"  { REDGE }
| "=="   { EQ }
| "!="   { NEQ }
| "<"    { LT }
| "<="   { LEQ }
| ">"    { GT }
| ">="   { GEQ }
| "def"  { DEF }
| "if"   { IF }
| "else" { ELSE }
| "true" { TRUE }
| "INF"  { INF }
| "false" { FALSE }
| "in"   { IN }
| "for"  { FOR }
| "while" { WHILE }
| "return" { RETURN }
| "bool" { BOOL }
| "num"  { NUM }
| "string" { STRING }
| "node" { NODE }
| "graph" { GRAPH }
| "list" { LIST }
| "dict" { DICT }
| num_regex as lxm { NUM_LIT(lxm) } (* num literal *)
| '"' ([^'"']* as lxm) '"' { STR_LIT(lxm) } (* string literals *)
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

and multicomment = parse
  "*/" { token lexbuf }
| _  { multicomment lexbuf }

and comment = parse
  "\n" { token lexbuf }
| _  { comment lexbuf }
```

Listing 16: scanner.mll

A.2 parser.mly

```
%{ open Ast %}

/* Punctuation Tokens */
%token SEMI COLON LPAREN RPAREN LBRACE RBRACE LBRACKET RBRACKET COMMA DOT
/* Arithmetic Operation Tokens */
%token PLUS MINUS TIMES DIVIDE
/* Assignment Operator */
%token ASSIGN
/* Comparative Operators */
%token EQ NEQ LT LEQ GT GEQ
/* Logical Operators */
%token LOGAND LOGOR
/* Node Operators */
%token UEDGE REDGE
/* Function Keyword Tokens */
%token RETURN IF ELSE FOR WHILE DEF IN
/* Punctuation Tokens */
%token BOOL NUM STRING NODE GRAPH LIST DICT
/* Boolean Operations */
%token TRUE FALSE INF

%token <string> NUM_LIT
%token <string> STR_LIT
%token <string> ID
%token EOF

/* Order of Operation */
%nonassoc NOCALL
%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left LOGOR
%left LOGAND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE

%start program               /* start symbol */
%type <Ast.program> program  /* return type program object */

%%

/* START PROGRAM */

program:
 decls EOF { $1 }

decls:
 | /* nothing */ { { cmds = [] } }
 | decls stmt { { cmds = $2 :: $1.cmds } }


////////////////////////////////////////////////////////////////////////////////
```

```
                /* FUNCTIONS */
////////////////////////////////////////////////////////////////////////////

/* (1)def (2)func (3)<funcName> ( (5)arg1,...argN ) { (8) <local variables>
    (9) <body> } */
fdecl:
  DEF f_data_type ID LPAREN formals_opt RPAREN LBRACE non_func_stmt_list
      RBRACE
    { Fdecl({
           rtype = $2;
           fname = $3;
           formals = $5;
           body = List.rev $8
   }) }

/* Optional Formal Args */
formals_opt:
   /* nothing */ { [] }
 | formal_list { List.rev $1 }

formal_list:
   f_data_type ID           { [($1, $2)] }
 | formal_list COMMA f_data_type ID { ($3, $4) :: $1 }

/* comma separated list of operations on nodes
 * for use with graph declarations
 */
 /*  Edge Operations  */
edge_op_list:
| edge_op { [$1] }
| edge_op_list COMMA edge_op { $3 :: $1 }

edge_op:
/* ID { NoOp($1) }*/
|  ID UEDGE ID { Undir($1, $3) }                 /*  x -- y      */
|  ID REDGE ID { Dir($1, $3) }                   /*  x --> y     */
|  ID REDGE LBRACKET expr RBRACKET ID { DirVal($1, $6, $4) } /* x -->[5] y */
|  ID UEDGE LBRACKET expr RBRACKET ID { UndirVal($1, $6, $4) } /* x --[5] y */
/* bug with bidirectional weighted edges: */
/*| ID LBRACKET expr RBRACKET UEDGE LBRACKET expr RBRACKET ID */ /* x [3]--[5]
     y */
  /*{ BidirVal($3, $1, $9, $7) }                 */

////////////////////////////////////////////////////////////////////////////
                    /* VARIABLES */
////////////////////////////////////////////////////////////////////////////

/* Literals */
literal:
| NUM_LIT { NumLiteral($1) }
| STR_LIT { StrLiteral($1) }
| list_literal {$1}
| dict_literal {$1}

```

```
list_literal:
| LBRACKET actuals_opt RBRACKET { ListLiteral($2) }

dict_literal:
| LBRACE tuples_opt RBRACE { DictLiteral($2)}


/* Primitive Typenames */
prim_type:
| BOOL { "bool" }
| NUM  { "num" }
| STRING { "string" }

data_type:
| prim_type { $1 }
| DICT LT data_type COMMA data_type GT { "dict" }
| LIST LT data_type GT { "list" }
| NODE { "node" }
| GRAPH { "graph" }

f_data_type:
| prim_type { Basic($1) }
| DICT LT data_type COMMA data_type GT { Dict($3,$5) }
| LIST LT data_type GT { List($3) }
| NODE { Basic("node") }
| GRAPH { Basic("graph") }

vdecl:
| prim_decl_prefix SEMI { $1 }
| node_decl_prefix SEMI { $1 }
| graph_decl_prefix SEMI { $1 }
| list_decl_prefix SEMI { $1 }
| dict_decl_prefix SEMI { $1 }

/* PRIMITIVE INITIALIZERS */
prim_decl_prefix:
| prim_type ID { Vdecl($1, $2) }                               /* num x */
| prim_type ID ASSIGN expr { Block([Vdecl($1, $2); Assign(Id($2), $4)]) } /*
   MOVE THESE */

/* NODE INITIALIZERS */
node_decl_prefix:
| NODE ID { Block[Vdecl("node", $2); NodeDef($2, Noexpr)] }
                              /* node x; */
| NODE ID LPAREN expr RPAREN { Block([Vdecl("node", $2); NodeDef($2, $4)]) }
   /* node x("chicago") */ /* node x("Chicago") */

/* GRAPH INITIALIZERS */
graph_decl_prefix:
| GRAPH ID { Vdecl("graph", $2) }                                 /* graph g;
   */
/*| GRAPH ID ASSIGN LBRACE edge_op_list RBRACE { Block([Vdecl("graph", $2);
   AssignList($2, $5)]) } */ /* graph g = { x --[5] y; y -->[3] z; } */
| GRAPH ID ASSIGN LBRACE edge_op_list RBRACE { Block([Vdecl("graph", $2);
   GraphDef($2,$5)]) } /* graph g = { x --[5] y; y -->[3] z; } */


list_decl_prefix:
| LIST LT data_type GT ID { ListDecl($3, $5) }
                                  /* list<node> min; */
| LIST LT data_type GT ID ASSIGN expr { Block([ListDecl($3, $5); Assign(Id($5)
   , $7)]) } /* list<node> min_path = { x, y, z; }; */

dict_decl_prefix:
| DICT LT data_type COMMA data_type GT ID { DictDecl($3, $5, $7) }
                 /* dict<node, num> parents; */
| DICT LT data_type COMMA data_type GT ID ASSIGN expr { Block([DictDecl($3, $5
   , $7); Assign(Id($7), $9)]) } /* dict<node, num> parents = { x; y; z; };
   */


////////////////////////////////////////////////////////////////////////////////
                          /* STATEMENTS */
////////////////////////////////////////////////////////////////////////////////

/* statements are defined inside functions or executed like a script */
/* a statement is just an action. ex. x = 5; */

stmt:
 | non_func_stmt {$1}
 | func_stmt {$1}

non_func_stmt:
 | expr SEMI { Expr($1) }
 | log_expr SEMI { Expr($1) }
 | edge_op SEMI { Expr($1) }
 | ID ASSIGN expr SEMI { Assign(Id($1), $3) }
 /*| access_expr ASSIGN expr SEMI { AccessAssign($1, $3) } */
 | expr LBRACKET expr RBRACKET ASSIGN expr SEMI { AccessAssign($1, $3, $6) }
 | RETURN expr SEMI { Return($2) }
 /* | LBRACE stmt_list RBRACE { Block(List.rev $2) } */
 | IF LPAREN log_expr RPAREN LBRACE stmt_list RBRACE %prec NOELSE { If($3,
    Block($6), Block([])) }
 | IF LPAREN log_expr RPAREN LBRACE stmt_list RBRACE ELSE LBRACE stmt_list
    RBRACE { If($3, Block($6), Block($10)) }
 | FOR LPAREN ID IN expr RPAREN LBRACE stmt_list RBRACE
   { For($3, $5, $8) }
 | WHILE LPAREN log_expr RPAREN LBRACE stmt_list RBRACE { While($3, $6) }
 | vdecl { $1 }

func_stmt:
   | fdecl {$1}

/* list of statements */
stmt_list:
   /* nothing */ { [] }
 | stmt_list stmt { $2 :: $1 }

```

```
non_func_stmt_list:
 /* nothing */ { [] }
 | non_func_stmt_list non_func_stmt { $2 :: $1 }

////////////////////////////////////////////////////////////////////////////
                    /* EXPRESSIONS */
////////////////////////////////////////////////////////////////////////////


log_expr:
 | expr EQ expr { Binop($1, Equal, $3) }
 | expr NEQ expr { Binop($1, Neq, $3) }
 | expr LT expr { Binop($1, Less, $3) }
 | expr LEQ expr { Binop($1, Leq, $3) }
 | expr GT expr { Binop($1, Greater, $3) }
 | expr GEQ expr { Binop($1, Geq, $3) }
 | log_expr LOGAND log_expr { Binop($1, LogAnd, $3) }
 | log_expr LOGOR log_expr { Binop($1, LogOr, $3) }

expr:
 | access_expr { $1 }
 | nacc_expr { $1 }

nacc_expr: /* non access exprs */
 | expr DOT ID LPAREN actuals_opt RPAREN { MemberCall($1, $3, $5) }
 | LPAREN expr RPAREN { $2 }
 | term          { $1 }

access_expr:
 | expr LBRACKET expr RBRACKET { Access($1, $3) }

term :
 | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
 | term PLUS term { Binop($1, Add, $3) }
 | term MINUS term { Binop($1, Sub, $3) }
 | term TIMES term { Binop($1, Mult, $3) }
 | term DIVIDE term { Binop($1, Div, $3) }
 | atom         { $1 }

atom:
 literal      { $1 }
 | INF          { NumLiteral("INF") }
 | TRUE         { Boolean(True) }
 | FALSE        { Boolean(False) }
 | ID           { Id($1) }


////////////////////////////////////////////////////////////////////////////
                       /* actuals */
////////////////////////////////////////////////////////////////////////////
actuals_opt:
   /* nothing */ { [] }
 | actuals_list { List.rev $1 }

```

```
tuples_opt:
 /* nothing*/ {[]}
 | tuples_list {List.rev $1}

/* for dictionary assignment */
tuples_list:
   expr COLON expr { [($1, $3)] }
  | tuples_list COMMA expr COLON expr { ($3, $5) :: $1 }

/* arguments to a function */
actuals_list:
   expr                { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }
```

Listing 17: parser.mly

A.3 ast.ml

```
type op = | Add | Sub | Mult | Div
        | Equal | Neq | Less | Leq | Greater | Geq
        | LogAnd (* && *)
        | LogOr (* || *)

type bool = True | False

type fun_dt =
  | Basic of (string)
  | List of (string)
  | Dict of (string * string)

type expr =
   NumLiteral of string
  | StrLiteral of string
  | ListLiteral of expr list (* ex. [1, 3, 42.33] *)
  | DictLiteral of (expr * expr) list (* ex. [(key, value)] *)
  | Boolean of bool
  | Id of string
  | Binop of expr * op * expr
  | Call of string * expr list
  | Access of expr * expr (* for dict and list element access, node.in[node2]
    *)
  | MemberCall of expr * string * expr list (* expr that evaluates to parent
    variable, accessed funct, parameters *)
  | Undir of string * string (* id, id *)
  | Dir of string * string (* id, id *)
  | UndirVal of string * string * expr (* id, id, weight *)
  | DirVal of string * string * expr (* id, id, weight *)
  | BidirVal of expr * string * string * expr (* weight, id, id, weight *)
  | NoOp of string
  | Noexpr


type stmt =
   Block of stmt list
  | Expr of expr
  | Vdecl of string * string (* (type, id) *)
  | ListDecl of string * string (* elem_type, id *)
  | DictDecl of string * string * string (* key_type, elem_type, id *)
  | Assign of expr * expr
  | AccessAssign of expr * expr * expr (* a[5] = 10 where first expr is an
     access expr *)
  | NodeDef of string * expr (* (node id, what goes inside parens) of item *)
(*  | AssignList of string * expr *)(* when a list of expressions is assigned
    to a variable *)
  | GraphDef of string * expr list (* id EdgeOp list - in form of undir dir -
     *)
  | Return of expr
  | If of expr * stmt * stmt
  | For of string * expr * stmt list (* temp var, iterable var, var decls,
     stmts *)
  | While of expr * stmt list (* condition, var decls, stmt list *)
  | Fdecl of func_decl and

   func_decl = {
    rtype : fun_dt;
    fname : string;
    formals : (fun_dt * string) list;
    (*locals : string list;*)
    body : stmt list;
  }

type program = { cmds: stmt list }

(*
    ///////////////////////////////////////////////////////////////////////////

                          /* PRETTY PRINTER */
///////////////////////////////////////////////////////////////////////////
    *)

(* prepends prelist at the head of postlst *)
let rec base_concat postlst = function
  | [] -> postlst
  | hd :: tl -> base_concat (hd :: postlst) tl

let concat prelst postlst = base_concat postlst (List.rev prelst)

let rec string_of_expr = function
    NumLiteral(l) -> l
  | StrLiteral(l) -> "\"" ^ l ^ "\""
  | ListLiteral(el) -> "[" ^ String.concat "," (List.map string_of_expr el) ^
     "]"
  | DictLiteral(el) -> "[" ^ String.concat "," (List.map (fun f -> "(" ^ (
     string_of_expr (fst f)) ^ " : " ^ (string_of_expr (snd f)) ^ ")" ) el)
  | Boolean(b) -> if b = True then "true" else "false"
  | Id(s) -> s
  | Binop(e1, o, e2) ->
     string_of_expr e1 ^ " " ^
    (match o with
        Add -> "+"
       | Sub -> "-"
       | Mult -> "*"
       | Div -> "/"
       | Equal -> "=="
       | Neq -> "!="
       | Less -> "<"
       | Leq -> "<="
       | Greater -> ">"
       | Geq -> ">="
       | LogAnd -> "&&"
       | LogOr -> "||"
     ) ^ " " ^

    string_of_expr e2
  | Undir (s1, s2) -> s1 ^ " -- " ^ s2
  | Dir (s1, s2) -> s1 ^ " --> " ^ s2
  | UndirVal (s1, s2, w) -> s1 ^ " --[" ^ string_of_expr w ^ "] " ^ s2
  | DirVal (s1, s2, w) -> s1 ^ " -->[" ^ string_of_expr w ^ "] " ^ s2
  | BidirVal (w1, s1, s2, w2) -> s1 ^ " [" ^ string_of_expr w1 ^ "]--[" ^
     string_of_expr w2 ^ "] " ^ s2
  | NoOp (s) -> s
  | Call(f, el) ->
    f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Access (e, e1) -> string_of_expr e ^ "[" ^ string_of_expr e1 ^ "]"
  | MemberCall (e1, s2, el) -> string_of_expr e1 ^ "." ^ s2 ^ "(" ^ String.
     concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""

let rec f_type_to_string = function
  | Basic(t) -> t
  | List(t) -> "list <" ^ t ^ ">"
  | Dict(kt,vt) -> "dict <" ^ kt ^ "," ^ vt ^ ">"

let rec string_of_stmt = function
   Block(stmts) ->
    "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
  | Expr(expr) -> string_of_expr expr ^ ";\n";
  | Vdecl(dt, id) -> dt ^ " " ^ id ^ ";\n";
  | ListDecl(dt, id) -> "list <" ^ dt ^ "> " ^ id ^ ";\n"
  | DictDecl(kdt, vdt, id) -> "dict <" ^ kdt ^ ", " ^ vdt ^ "> " ^ id ^ ";\n"
  | Assign(v, e) -> string_of_expr v ^ " = " ^ string_of_expr e ^ ";"
  | AccessAssign(e1, e2, e3) -> string_of_expr e1 ^ "[" ^ string_of_expr e2 ^
     "] = " ^ string_of_expr e3 ^ ";\n"
  | NodeDef(v, e) -> v ^ "(" ^ string_of_expr e ^ ")" (* (node id, what goes
     inside parens) of item *)
  | GraphDef(v, el) -> v ^ " = { " ^ String.concat "," (List.map
     string_of_expr el) ^ "};"
  | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt
     s
  | If(e, s1, s2) -> "if (" ^ string_of_expr e ^ ")\n" ^
    string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | For(e1, e2, sl) ->
     "for (" ^ e1 ^ " in " ^ string_of_expr e2
     ^ ") { " ^ String.concat "\n" (List.map string_of_stmt sl) ^ " }"
  | While(e, sl) -> "while (" ^ string_of_expr e ^ ") {" ^ String.concat "\n"
      (List.map string_of_stmt sl) ^ " }"
  | Fdecl(f) -> string_of_fdecl f and

  string_of_fdecl fdecl =
  "def " ^ (f_type_to_string fdecl.rtype) ^ " " ^ fdecl.fname ^ "(" ^
    (String.concat ", " (List.map (fun f ->(f_type_to_string (fst f)) ^ " " ^
       snd f) fdecl.formals)) ^
    ")\n{\n" ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_vdecl id = "type " ^ id ^ ";\n"


let string_of_program (funcs, cmds) =
  String.concat "\n" (List.map string_of_fdecl funcs) ^
  String.concat "\n" (List.map string_of_stmt cmds)
```
Listing 18: ast.ml

A.4 Sast.ml

```
(* this defines semantically typed dots ast *)
module StringMap = Map.Make(String)

type dataType = | Num | String | Bool
             | Graph | Node
             | List of dataType (* val type *)
             | Dict of dataType * dataType (* key type, val type *)
             | Void



type s_expr =
   NumLiteral of string * dataType             (* 5 *)
  | StrLiteral of string * dataType            (* "Hello" *)
  | ListLiteral of s_expr list * dataType      (* [2.5, 3, x] *)
  | DictLiteral of (s_expr * s_expr) list * dataType (* [(Hello, 15)] *)
  | Boolean of Ast.bool * dataType             (* True *)
  | Id of string * dataType                    (* x *)
  | Binop of s_expr * Ast.op * s_expr * dataType (* x + y *)
  | Call of string * s_expr list * dataType
  | Access of s_expr * s_expr * dataType       (* for dict and list element
     access *)
  | MemberCall of s_expr * string * s_expr list * dataType (* parent variable,
      accessed funct, parameters *)
  | Undir of string * string * dataType        (* id, id *)
  | Dir of string * string * dataType          (* id, id *)
  | UndirVal of string * string * s_expr * dataType (* id, id, weight *)
  | DirVal of string * string * s_expr * dataType (* id, id, weight *)
  | BidirVal of s_expr * string * string * s_expr * dataType (* weight, id, id
    , weight *)
  | NoOp of string * dataType
  | Noexpr

type s_stmt =
   Block of s_stmt list
  | Expr of s_expr
  | Vdecl of dataType * string
  | NodeDef of string * s_expr * dataType (* (node id, item id, datatype) *)
  | GraphDef of string * s_expr list
  | Assign of s_expr * s_expr * dataType      (* x = 5; *)
  | AccessAssign of s_expr * s_expr * s_expr * dataType (* a[5] = 10 where
    first thing is an access expr *)
  | Return of s_expr * dataType            (* return x (dataType) *)
  | If of s_expr * s_stmt * s_stmt  (* if (boolean) stmt; *)
  | For of string * s_expr * s_stmt list (* temp var, iterable var, stmts *)
  | While of s_expr * s_stmt list   (* condition, var decls, stmt list *)
  | Fdecl of s_fdecl and

 s_fdecl = {
   s_fname : string;
   s_rtype : dataType;
   s_formals : (dataType * string) list;
   s_body : s_stmt list;
  }

type program = { s_cmds : s_stmt list }

let rec dt_to_str = function
| Num -> "num"
| String -> "string"
| Bool -> "bool"
| List(dt) -> "list<" ^ (dt_to_str dt) ^ ">"
| Dict(dtk, dtv) -> "dict<" ^ (dt_to_str dtk) ^ ", " ^ (dt_to_str dtv) ^ ">"
| Graph -> "graph"
| Node -> "node"
| Void -> "void"

(* end Sast *)
```

Listing 19: Sast.ml

A.5 typeConverter.ml

```
(*****************************)
(* CONVERTS AN AST TO AN SAST *)
(******************************)
open Ast
open Sast
open Analyzer

(* extract dt from list *)
let get_list_type = function
```

```
 | Sast.List(dt) -> dt
 | _ -> raise (Failure("wrong type: not a list"))

(* extract key type, val type from dict *)
let get_dict_type = function
 | Sast.Dict(dt1, dt2) -> (dt1, dt2)
 | _ -> raise (Failure("wrong type: not a dict "))

(* make sure each element in a list is the right type *)
let rec check_list env v_e = function
 | [] -> ""
 | hd::tl ->
   if not(v_e = (get_sexpr_type hd)) then
     raise (Failure ("list element not of type: " ^ type_to_str ( v_e )) )
   else
    check_list env v_e tl


let rec check_graph_list env = function
 | [] -> ""
 | hd::tl ->
     (match hd with
      | Sast.Id(v, dt) ->
        let e_dt = get_sexpr_type hd in
        (if not (e_dt = Sast.Node || e_dt = Sast.Graph) then
          raise (Failure ("you can not have a graph def with type other than
              Node or Graph"))
        else
          check_graph_list env tl)
      | Sast.Undir(v1,v2,dt) | Sast.Dir(v1,v2,dt) ->
          check_graph_list env tl
      | Sast.UndirVal(v1,v2,e1,dt) | Sast.DirVal(v1,v2,e1,dt) ->
         check_graph_list env tl
      | Sast.BidirVal(e1,v1,v2,e2,dt) ->
          check_graph_list env tl
      | _ -> raise (Failure ("type not expected in Graph Def")))

(* make sure each pair in dict assignment is right type *)
let rec check_dict env v_e = function
 | [] -> ""
 | hd::tl ->
   if not((fst v_e = get_sexpr_type (fst hd)) && (snd v_e = get_sexpr_type (
       snd hd))) then
     raise (Failure ("assignment expression not of type: " ) )
   else
     check_dict env v_e tl

(* match arguments to a function call to that func's definition *)
 let rec formal_check s_formal_list s_el =
  match s_formal_list, s_el with
   | [], [] -> true
   | [], _ -> raise (Failure ("incorrect arguments" ) )
   | _, [] -> raise (Failure ("incorrect arguments" ) )
   | hd1::tl1, hd2::tl2 -> ((fst hd1) = (get_sexpr_type hd2)) && (formal_check
      tl1 tl2)


(* converts Ast.program to Sast.program *)
let convert_ast prog env =

(* convert an Ast.expr object to Sast.expr object *)
let rec expr env = function
| Ast.NumLiteral(v) -> Sast.NumLiteral(v, Sast.Num)
| Ast.StrLiteral(v) -> Sast.StrLiteral(v, Sast.String)
| Ast.ListLiteral(el) ->
  let s_el = List.map (expr env) el in
  (match el with
   | [] -> ListLiteral([], List(Void))
   | x -> let dt = get_sexpr_type (List.hd s_el)
         in
         ignore (check_list env dt s_el);
         ListLiteral(s_el, List(dt))
  )
| Ast.DictLiteral(el) ->
(* key_type, elem_type, id [(Hello, 15)] *)
  let s_el = List.map (fun f -> (expr env (fst f), expr env (snd f))) el in
  (match s_el with
    | [] -> DictLiteral([], Dict(Void,Void))
    | x ->
     let dt = (get_sexpr_type (fst(List.hd s_el)), get_sexpr_type (snd(List.hd
         s_el)))
          in
          ignore (check_dict env dt s_el);
          DictLiteral(s_el, Sast.Dict(fst dt, snd dt))
  )
| Ast.Boolean(b) -> Sast.Boolean(b, Sast.Bool)
| Ast.Id(v) ->
    (try
       Sast.Id(v, find_var v env.var_types) (* uses find_var to determine the
           type of id *)
     with
     | Not_found -> raise (Failure ("undeclared variable: " ^ v))
    )
| Ast.Binop(e1, op, e2) ->
    let s_e1 = expr env e1 in
    let s_e2 = expr env e2 in
    let e1_dt = get_sexpr_type s_e1 in
    let e2_dt = get_sexpr_type s_e2 in
    (match op with
    | Add ->
      (match e1_dt with
        | Num ->
          (match e2_dt with
            | Num -> Sast.Binop(s_e1,op,s_e2,Sast.Num)
            | String -> Sast.Binop(s_e1,op,s_e2,Sast.String)
            | _ -> raise (Failure("wrong type: Num + ? "))
          )
        | String ->
```

```
          (match e2_dt with
            | Num -> Sast.Binop(s_e1,op,s_e2,Sast.String)
            | String -> Sast.Binop(s_e1,op,s_e2,Sast.String)
            | _ -> raise (Failure("wrong type: String + ? "))
          )
      | Graph ->
          (match e2_dt with
            | Node -> Sast.Binop(s_e1,op,s_e2,Sast.Graph)
            | Graph -> Sast.Binop(s_e1,op,s_e2,Sast.Graph)
            | _ -> raise (Failure("wrong type: Graph + ? "))
          )
      | Node ->
          (match e2_dt with
            | Node -> Sast.Binop(s_e1,op,s_e2,Sast.Graph)
            | Graph -> Sast.Binop(s_e1,op,s_e2,Sast.Graph)
            | _ -> raise (Failure("wrong type: Node + ? "))
          )
      | List(dt) ->
          (match e2_dt with
            | List(dt) ->
                if (e1_dt = e2_dt) then
                  Sast.Binop(s_e1,op,s_e2,e1_dt)
                else
                  raise (Failure("wrong type: List + List<?> "))
            | _ -> raise (Failure("wrong type: List + ? "))
          )
      | _ -> raise (Failure("Expr using + has incompatible types"))
    )
  | Sub ->
    (match e1_dt with
      | Num ->
          (match e2_dt with
            | Num -> Sast.Binop(s_e1,op,s_e2,Sast.Num)
            | _ -> raise (Failure("wrong type: Num - ? "))
          )
      | Graph ->
          (match e2_dt with
            | Node -> Sast.Binop(s_e1,op,s_e2,Sast.Graph)
            | Graph -> Sast.Binop(s_e1,op,s_e2,Sast.Graph)
            | _ -> raise (Failure("wrong type: Graph - ? "))
          )
      | _ -> raise (Failure("Expr using - has incompatible types"))
    )
  | Mult | Div ->
    (match e1_dt with
      | Num ->
          (match e2_dt with
            | Num -> Sast.Binop(s_e1,op,s_e2,Sast.Num)
            | _ -> raise (Failure("wrong type: Num * or / ? "))
          )
      | _ -> raise (Failure("Expr using / has incompatible types"))
    )
  | Equal | Neq ->
    (match e1_dt with
      | Num ->
          (match e2_dt with
            | Num -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: Num ==/!= ? "))
          )
      | String ->
          (match e2_dt with
            | String -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: String ==/!= ? "))
          )
      | Bool ->
          (match e2_dt with
            | Bool -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: Bool ==/!= ? "))
          )
      | Void ->
          (match e2_dt with
            | Void -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: Void ==/!= ? "))
          )
      | Graph ->
          (match e2_dt with
            | Node -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | Graph -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: Graph ==/!= ? "))
          )
      | Node ->
          (match e2_dt with
            | Node -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | Graph -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: Node ==/!= ? "))
          )
      | List(dt) ->
          (match e2_dt with
            | List(dt) ->
                if (e1_dt = e2_dt) then
                  Sast.Binop(s_e1,op,s_e2,Sast.Bool)
                else
                  raise (Failure("wrong type: List ==/!= List<?> "))
            | _ -> raise (Failure("wrong type: List ==/!= ? "))
          )
      | Dict(dtk,dtv) ->
          (match e2_dt with
            | Dict(dtk,dtv) ->
                if (e1_dt = e2_dt) then
                  Sast.Binop(s_e1,op,s_e2,Sast.Bool)
                else
                  raise (Failure("wrong type binop: Dict ==/!= Dict<?> "))
            | _ -> raise (Failure("wrong type: Dict ==/!= ? "))
          )
    )

  | Less | Leq | Greater | Geq ->
    (match e1_dt with
        | Num ->
          (match e2_dt with
            | Num -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: Num </>/<=/>= ? "))
          )
        | String ->
          (match e2_dt with
            | String -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: String </>/<=/>= ? "))
          )
        | _ -> raise (Failure("Expr using </>/<=/>= has incompatible types"))
      )

    | LogAnd | LogOr ->
      (match e1_dt with
        | Bool ->
          (match e2_dt with
            | Bool -> Sast.Binop(s_e1,op,s_e2,Sast.Bool)
            | _ -> raise (Failure("wrong type: Bool &&/|| ? "))
          )
        | _ -> raise (Failure("Expr using &&/|| has incompatible types"))
      )
  )
| Ast.Call(f, el) ->
    let s_el = List.map (expr env) el in
  (*
  match range can only take 1 or two args not 0 or 3+ and make sure nums...
  instead of f try check with the func name
  if it does exist put the value of the key function name for the map
      func_types
  s_formals : (dataType * string) list;
  range(1,5)
    [1,2,3,4,5]
   *)
  if f = "range" then
      (let len = List.length el in
      if (len = 1) then
        let arg_types = [(Sast.Num, "foo")] in
        ignore (formal_check arg_types s_el);
        Sast.Call(f, s_el , Sast.List(Sast.Num))
      else if (len = 2) then
        let arg_types = [(Sast.Num, "foo"); (Sast.Num, "foo")] in
        ignore (formal_check arg_types s_el);
        Sast.Call(f, s_el , Sast.List(Sast.Num))
      else
        raise(Failure("range can only take 1 or 2 args"))
    )
  else if f = "print" then Sast.Call(f, s_el, Sast.Void)
  else if (f = "min" || f = "max") then
      (try
        let s_el = List.map (expr env) el in
        let data_type = get_sexpr_type (List.hd s_el) in
        let len = List.length el in
          if (len = 1) then
```

```
            (match data_type with
              | Sast.List(dt) -> Sast.Call(f, s_el, dt)
              | Sast.Dict(dtk, dtv) -> Sast.Call(f, s_el, dtk)
              | _ -> raise(Failure("member call failed")))
         else
           raise(Failure("member call failed"))
       with
         Not_found -> raise (Failure("undeclared variable: ")))
  else if f = "len" then
       (try
       let s_el = List.map (expr env) el in
       let data_type = get_sexpr_type (List.hd s_el) in
       let len = List.length el in
         if (len = 1) then
            (match data_type with
              Sast.List(dt) -> Sast.Call(f, s_el, Sast.Num)
              | Sast.Dict(dtk, dtv) -> Sast.Call(f, s_el, Sast.Num)
              | _ -> raise(Failure("member call failed")))
         else
           raise(Failure("member call failed"))
       with
         Not_found -> raise (Failure("undeclared variable: ")))
  else
    (
     let fdecl = find_var f env.func_obj in
      ignore (formal_check fdecl.s_formals s_el);
      let rtype = fdecl.s_rtype in
      Sast.Call(f, s_el , rtype)
    )
| Ast.Access(e1, e2) ->
    let s_e1 = expr env e1 in    (* func rec until it knows datatype -- sast
        version of ast expr e *)
    let e1_dt = get_sexpr_type s_e1 in
    let s_e2 = expr env e2 in    (* func rec until it knows datatype -- sast
        version of ast expr e *)
    let e2_dt = get_sexpr_type s_e2 in
    (try                          (*sees if variable defined*)
        (match e1_dt with
         List(dt) ->
            (match e2_dt with
              | Sast.Num -> Sast.Access(s_e1, s_e2, dt)
              | _ -> raise (Failure "expr to access list should be Num")
            )
        | Dict(dk,dv) ->
          if (e2_dt = dk) then
              Sast.Access(s_e1, s_e2, dv)
          else
              raise (Failure("wrong type ast.access: sexpr type != Dict<?>,
                  expected " ^ (type_to_str dk) ^ " got " ^ (type_to_str e2_dt)
                  ^ " life sucks"))
        | _ -> raise (Failure("must use Dict or List with access!"))
        )
     with
     | Not_found -> raise (Failure("undeclared variable: "))
```

```
   );
| Ast.MemberCall(e, m, el) ->
(
   let s_e = expr env e in
   let e_dt = get_sexpr_type s_e in
   let num_args = List.length el in
   let s_el = List.map (expr env) el in

   match m with
   | "enqueue" | "push" ->
      if num_args != 1 then raise (Failure ("enqueue/push requires 1 arg"))
      else
         ignore((match e_dt with
          | List(_) -> ignore()
          | _ -> raise (Failure ("enqueue/push error: not a list"))
         ));
         ignore(check_list env (get_list_type e_dt) s_el); (* check that the
            arg is the type in the list *)
         Sast.MemberCall(s_e, m, s_el, Sast.Void)
   | "dequeue" | "pop" ->
      if num_args != 0 then raise (Failure ("dequeue/pop requires 0 args"))
      else Sast.MemberCall(s_e, m, s_el, Sast.Void)
   | "peek" ->
      if num_args != 0 then raise (Failure ("peek requires 0 args"))
      else Sast.MemberCall(s_e, m, s_el, get_list_type e_dt)
   | "oute" | "ine" ->
      if num_args != 0 then raise (Failure ("oute/ine requires 0 args"))
      else Sast.MemberCall(s_e, m, s_el, Dict(Node, Num))
   | "val" ->
      if num_args != 0 then raise (Failure ("val requires 0 args"))
      else Sast.MemberCall(s_e, m, s_el, String)
   | "remove" ->
      if num_args != 1 then raise (Failure ("remove requires 1 arg"))
      else
         ignore((match e_dt with
          | Dict(_) -> ignore()
          | _ -> raise (Failure ("remove error: not a dict"))
         ));
         ignore(check_list env (fst (get_dict_type e_dt)) s_el); (* check that
             the arg is the type in the list *)
         Sast.MemberCall(s_e, m, s_el, Sast.Void)
   | _ -> raise (Failure ("no member function: " ^ m))
)
| Ast.Undir(v1, v2) ->
   (*check if v1 and v2 exist *)
   (try                          (*sees if variable defined*)
      let v1_e = find_var v1 env.var_types in
       if v1_e = Sast.Node then
         let v2_e = find_var v2 env.var_types in
          if v2_e = Sast.Node then
            Sast.Undir(v1, v2, Sast.Void)
          else raise (Failure("Wrong variable types"))
        else
         raise (Failure("Wrong variable types"))
```

```
    with
    | Not_found -> raise (Failure("undeclared variable: ")))

| Ast.Dir(v1, v2) ->
  (try                           (*sees if variable defined*)
     let v1_e = find_var v1 env.var_types in
       if v1_e = Sast.Node then
         let v2_e = find_var v2 env.var_types in
           if v2_e = Sast.Node then
             Sast.Dir(v1, v2, Sast.Void)
           else raise (Failure("undeclared variable: "))
       else raise (Failure("undeclared variable: "))
    with
    | Not_found -> raise (Failure("undeclared variable: ")))
| Ast.BidirVal(w1, v1, v2, w2) ->
    (try                           (*sees if variable defined*)
        if find_var v1 env.var_types = Sast.Graph then
            if find_var v2 env.var_types = Sast.Graph then
              let s_w1 = expr env w1 in
              if get_sexpr_type s_w1 = Sast.Num then
                let s_w2 = expr env w2 in
                if get_sexpr_type s_w2 = Sast.Num then
                  Sast.BidirVal(expr env w1, v1, v2, expr env w2, Sast.Void)
                else
                  raise (Failure("undeclared variable: "))
              else
                raise (Failure("undeclared variable: "))
            else
              raise (Failure("undeclared variable: "))
        else
          raise (Failure("undeclared variable: "))
    with
    | Not_found -> raise (Failure("undeclared variable: ")))
| Ast.UndirVal(v1, v2, w) ->
     (try                           (*sees if variable defined*)
          if find_var v1 env.var_types = Sast.Node then
              if find_var v2 env.var_types = Sast.Node then
                let s_w = expr env w in
                if get_sexpr_type s_w = Sast.Num then
                    Sast.UndirVal(v1, v2, expr env w, Sast.Void)
                else
                  raise (Failure("undeclared variable: "))
              else
                raise (Failure("undeclared variable: "))
          else
            raise (Failure("undeclared variable: "))
    with
       | Not_found -> raise (Failure("undeclared variable: ")))
| Ast.DirVal(v1, v2, w) ->
   (try
      if find_var v1 env.var_types = Sast.Node then
        if find_var v2 env.var_types = Sast.Node then
          let s_w = expr env w in
          if get_sexpr_type s_w = Sast.Num then
```

```
            Sast.DirVal(v1, v2, expr env w, Sast.Void)
          else
            raise (Failure("undeclared variable: "))
        else
          raise (Failure("undeclared variable: "))
      else
        raise (Failure("undeclared variable: "))
     with
       | Not_found -> raise (Failure("undeclared variable: ")))
| Ast.NoOp(v) -> Sast.NoOp(v, Sast.Void)
| Ast.Noexpr -> Sast.Noexpr
in

(* convert an Ast.stmt object to Sast.stmt object *)
let rec stmt env = function
| Ast.Block(sl) -> Sast.Block(List.map (fun s -> stmt env s) sl)
| Ast.Expr(e) -> Sast.Expr(expr env e)
| Ast.Vdecl(dt, id) ->
    (try
       ignore (StringMap.find id !(List.hd env.var_types));
       raise (Failure ("variable already declared in local scope: " ^ id))
     with | Not_found -> (List.hd env.var_types) := StringMap.add id (
        str_to_type dt) !(List.hd env.var_types); (* add type map *)
             (List.hd env.var_inds) := StringMap.add id (find_max_index !(List.
                hd env.var_inds)+1) !(List.hd env.var_inds); (* add index
                mapping *)
         | Failure(f) -> raise (Failure (f) )
    );
    Sast.Vdecl(str_to_type dt, id)
| Ast.ListDecl(dt, id) -> (* Sast.ListDecl(str_to_type dt, v) *)
    let vtype = Sast.List(str_to_type dt) in
    (try
       ignore (StringMap.find id !(List.hd env.var_types));
       raise (Failure ("variable already declared in local scope: " ^ id))
     with | Not_found -> (List.hd env.var_types) := StringMap.add id vtype !(
        List.hd env.var_types); (* add type map *)
             (List.hd env.var_inds) := StringMap.add id (find_max_index !(List.
                hd env.var_inds)+1) !(List.hd env.var_inds); (* add index
                mapping *)
         | Failure(f) -> raise (Failure (f))
    );
    Sast.Vdecl(vtype, id)
| Ast.DictDecl(dtk, dtv, id) -> (*Sast.DictDecl(str_to_type dtk, str_to_type
    dtv, v)*)
    let vtype = Sast.Dict(str_to_type dtk, str_to_type dtv) in
    (try
       ignore (StringMap.find id !(List.hd env.var_types));
       raise (Failure ("variable already declared in local scope: " ^ id))
     with | Not_found -> (List.hd env.var_types) := StringMap.add id vtype !(
        List.hd env.var_types);
             (List.hd env.var_inds) := StringMap.add id (find_max_index !(List.
                hd env.var_inds)+1) !(List.hd env.var_inds);
         | Failure(f) -> raise (Failure (f) )
    );
```

```
   Sast.Vdecl(vtype, id)
| Ast.Assign(v, e) ->           (* checks that the var and expression are of
   the same type, then converts to Sast.Assign *)
   let s_e = expr env e in    (* func rec until it knows datatype -- sast
      version of ast expr e *)
   let s_v = expr env v in
   let e_dt = get_sexpr_type s_e in (* data type of that sast expr with
      function get_sexpr_type*)
   let v_dt = get_sexpr_type s_v in
   if not (v_dt = e_dt)
   then raise (Failure ("assignment expression not of type: " ^ (type_to_str
      v_dt) ))
   else Sast.Assign(s_v, s_e, Sast.Void)
| Ast.AccessAssign(e1, e2, e3) ->
   let s_e1 = expr env e1 in    (* func rec until it knows datatype -- sast
      version of ast expr e *)
   let e1_dt = get_sexpr_type s_e1 in
   let s_e2 = expr env e2 in    (* func rec until it knows datatype -- sast
      version of ast expr e *)
   let e2_dt = get_sexpr_type s_e2 in
   let s_e3 = expr env e3 in
   let e3_dt = get_sexpr_type s_e3 in
   (try                         (*sees if variable defined*)
       (match e1_dt with
        List(dt) ->
           (match e2_dt with
             | Sast.Num ->
             if (e3_dt = dt) then
             Sast.AccessAssign(s_e1, s_e2, s_e3, Sast.Void)
            else
            raise (Failure("AccessAssign: Assigning wrong type"))

             | _ -> raise (Failure ("expr to access list should be Num"))
           )
        | Dict(dk,dv) ->
         if (e2_dt = dk) then
            if(e3_dt = dv) then
            Sast.AccessAssign(s_e1, s_e2, s_e3, Sast.Void)
          else
          raise(Failure("AccessAssign: mismatched Value data type"))
         else
            raise (Failure("wrong type accessassign: Dict != Dict<?> "))
        | _ -> raise (Failure("must use Dict or List with access!"))
        )
    with
    | Not_found -> raise (Failure("undeclared variable: "))
   );

| Ast.NodeDef(v, e) -> (* (node id, what goes inside parens) of item *)
   (try
     let v_e = (find_var v env.var_types) in
     let s_e = expr env e in
     let e_dt = get_sexpr_type s_e in
     if v_e = Sast.Node then
```

```
      Sast.NodeDef(v, s_e, e_dt)
    else raise (Failure ("Node Def failure"))
  with
    | Not_found -> raise (Failure("Node Def failure")))
| Ast.GraphDef(v, el) ->
  (try
    let s_el = List.map (expr env) el in
    ignore(check_graph_list env s_el);
    Sast.GraphDef(v, s_el)
  with
   Not_found -> raise (Failure ("GraphDef issue")))

| Ast.Return(e) ->
  (try
      let s_e = expr env e in
     (match get_sexpr_type s_e with
       | Sast.Node -> Sast.Return(s_e, Sast.Node)
       | Sast.Num -> Sast.Return(s_e, Sast.Num)
       | Sast.String -> Sast.Return(s_e, Sast.String)
       | Sast.Bool -> Sast.Return(s_e, Sast.Bool)
       | Sast.Graph -> Sast.Return(s_e, Sast.Graph)
       | Sast.List(s_dt) -> Sast.Return(s_e, Sast.List(s_dt))
       | Sast.Dict(dtk, dtv) -> Sast.Return(s_e, Sast.Dict(dtk, dtv))
       | Sast.Void -> Sast.Return(s_e, Sast.Void))
  with
   Not_found -> raise (Failure ("return issue")))
| Ast.If(cond, s1, s2) ->
  (try
    let s_cond = expr env cond in
    if (get_sexpr_type s_cond) = Sast.Bool then
      Sast.If(expr env cond, stmt env s1, stmt env s2)
    else
      raise (Failure ("if issue"))
  with
   Not_found -> raise (Failure ("return issue")))
| Ast.For(v, e, sl) ->
  (* iterable expr must have var which already has been been declared *)
  let s_e = expr env e in
  let e_dt = get_sexpr_type s_e in
  (match e_dt with
    | Sast.List(dt) ->
      (try
        ignore(find_var v env.var_inds);
        ignore(raise (Failure ("'" ^ v ^ "' has already been declared")))
       with
       | Not_found -> ignore()
       | Failure(f) -> raise (Failure f)
      );
      (* add the temp var to the symbol table *)
      (List.hd env.var_types) := StringMap.add v dt !(List.hd env.var_types);
          (* add type map *)
      (List.hd env.var_inds) := StringMap.add v (find_max_index !(List.hd env.
         var_inds)+1) !(List.hd env.var_inds); (* add index mapping *)
      Sast.For(v, s_e, List.map (fun s -> stmt env s) sl)
     | Dict(dtk, dtv) ->
      (try
       ignore(find_var v env.var_inds);
       ignore(raise (Failure ("'" ^ v ^ "' has already been declared")))
       with
       | Not_found -> ignore()
       | Failure(f) -> raise (Failure f)
       );
      (* add the temp var to the symbol table *)
      (List.hd env.var_types) := StringMap.add v dtk !(List.hd env.var_types);
          (* add type map *)
      (List.hd env.var_inds) := StringMap.add v (find_max_index !(List.hd env.
         var_inds)+1) !(List.hd env.var_inds); (* add index mapping *)
      Sast.For(v, s_e, List.map (fun s -> stmt env s) sl)
     | Graph ->
      (try
       ignore(find_var v env.var_inds);
       ignore(raise (Failure ("'" ^ v ^ "' has already been declared")))
       with
       | Not_found -> ignore()
       | Failure(f) -> raise (Failure f)
       );
      (* add the temp var to the symbol table *)
      (List.hd env.var_types) := StringMap.add v Sast.Node !(List.hd env.
         var_types); (* add type map *)
      (List.hd env.var_inds) := StringMap.add v (find_max_index !(List.hd env.
         var_inds)+1) !(List.hd env.var_inds); (* add index mapping *)
      Sast.For(v, s_e, List.map (fun s -> stmt env s) sl)

     | Node ->
      (try
       ignore(find_var v env.var_inds);
       ignore(raise (Failure ("'" ^ v ^ "' has already been declared")))
       with
       | Not_found -> ignore()
       | Failure(f) -> raise (Failure f)
       );
      (* add the temp var to the symbol table *)
      (List.hd env.var_types) := StringMap.add v Sast.Node !(List.hd env.
         var_types); (* add type map *)
      (List.hd env.var_inds) := StringMap.add v (find_max_index !(List.hd env.
         var_inds)+1) !(List.hd env.var_inds); (* add index mapping *)
      Sast.For(v, s_e, List.map (fun s -> stmt env s) sl)
     | _ -> raise(Failure("Trying to for loop an expr that doesnt return an
        iterable"))
  )

| Ast.While(cond, sl) ->
     (try
      let s_cond = expr env cond in
      if (get_sexpr_type s_cond) = Sast.Bool then
         Sast.While(expr env cond, List.map (fun s -> stmt env s) sl)
      else
       raise (Failure ("while issue"))
   with
    Not_found -> raise (Failure ("while issue")))
| Ast.Fdecl(func) -> (*Fdecl of func_decl and *)
   (try
     (* add formal variables to local scope variable maps *)
    let fname = func.fname in
    let formals = List.map (fun (dt, v) -> (f_dt_to_type dt, v)) func.formals
        in
    let rtype = f_dt_to_type func.rtype in

  (* add this function to symbol table *)
  let dummy_func_obj = {
                      Sast.s_fname = fname;
                      Sast.s_rtype = rtype;
                      Sast.s_formals = [];
                      Sast.s_body = []
                    }
  in
  (List.hd env.func_obj) := StringMap.add fname dummy_func_obj !(List.hd env.
      func_obj);
  (List.hd env.func_inds) := StringMap.add fname (find_max_index !(List.hd env
      .func_inds)+1) !(List.hd env.func_inds); (* add index map *)

  let func_env = {
        var_inds = ref StringMap.empty :: env.var_inds; (* var names to
            indices ex. x -> 1 so that we can just refer to it as v1 *)
        var_types = ref StringMap.empty :: env.var_types; (* maps a var name
            to its type ex. x -> num *)
        func_inds = env.func_inds;    (* func names to indices ex. x -> 1 so
            that we can just refer to it as f1 *)
        func_obj = env.func_obj;
        return_type = rtype;             (* what should the return type be of
            the current scope *)
  }
  in

  let rec fmls_adder env = function
  | [] -> ignore()
  | hd :: tl ->
      (List.hd env.var_types) := StringMap.add (snd hd) (fst hd) !(List.hd env.
          var_types);
      (List.hd env.var_inds) := StringMap.add (snd hd) (find_max_index !(List.
          hd env.var_inds)+1) !(List.hd env.var_inds); (* add index map *)
      ignore(fmls_adder env tl)
  in
  ignore(fmls_adder func_env formals);

 let populated_fdecl = {
                      Sast.s_fname = func.fname;
                      Sast.s_rtype = rtype;
                      Sast.s_formals = formals;
                      Sast.s_body = List.map (fun s -> stmt func_env s) func.body
                     }
 in
 (List.hd env.func_obj) := StringMap.add fname populated_fdecl !(List.hd env.
    func_obj); (* replace the dummy fobj with the evaluated one *)
 Sast.Fdecl(populated_fdecl)
  with
    Not_found -> raise (Failure ("fdecl issue")))
in

 { Sast.s_cmds = List.map (fun s -> stmt env s) prog.cmds }

(* get printf fmt string for Sast.dataType types *)
let dt_fmt = function
| Sast.Num -> "%f"
| Sast.String -> "%s"
| Sast.Bool -> "%d"
| Sast.Graph -> "" (* TODO *)
| Sast.Node -> "" (* TODO *)
| Sast.Dict(dtk, dtv) -> "" (* TODO *)
| Sast.List(dt) -> "" (* TODO *)
| Sast.Void -> "" (* TODO *)
```

Listing 20: typeConverter.ml

A.6 analyzer.ml

```
(* converts dots SAST to C AST *)
open Ast
open Sast
open Translate


module StringMap = Map.Make(String)
type s_program = { s_globals : s_stmt list; s_main: s_stmt list; s_funcs :
   s_fdecl list; }

(*
 DEALING WITH AUTOMATIC RESULT VARS:

 Step 1: After every "let x = ...." where "translate_expr env ..." is called,
        create a variable to hold the name of the result variable from that
            call
        ex. let e1_result = "v" ^ string_of_int (find_max_index !(List.hd env.
            var_inds)) in

 Step 2: After the end of all "translate_expr env ..." calls (i.e. when that
    function
        is no longer called), create a new auto_var to hold the result of the
            current
        function's translation.

        ex. let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in

 Step 3: Output a Block([]) that contains:
        a. a Vdecl object for result_var
        b. the C code that corresponds to the current expr's translation
        c. an Assign call that assigns the result of part b. to result_var

        ex. for if the result of e1 is the result of your expression
            Block([Vdecl(..., result_var);
                  c_e1;
                  Assign(Id(..., result_var), Id(..., e1_result))
                ])
*)

(* where symbol tables are stored *)
type translation_env = {
         var_inds : int StringMap.t ref list;  (* var names to indices ex. x -> 1 so that we can just refer to it as v1 *)
         var_types : Sast.dataType StringMap.t ref list; (* maps a var name to its type ex. x -> num *)
         func_inds : int StringMap.t ref list; (* func names to indices ex. x -> 1 so that we can just refer to it as f1 *)
         func_obj : Sast.s_fdecl StringMap.t ref list; (* maps a func name to its return type *)
         return_type : Sast.dataType;          (* what should the return type be of the current scope *)
}

(*
 @param *implicit* := list of Sast.stmts to sort
 @param sifted := a struct that contains the globals,
    regular stmts, and func decls that have been sorted so far
 *)
let get_list_type = function
 | Sast.List(dt) -> dt
 | _ -> raise (Failure("wrong type: not a list"))

 let get_dict_type = function
  | Sast.Dict(dtk, dtv) -> dtk
  | _ -> raise (Failure("wrong type: not a list"))

(*
 extract an SAST into a form more presentable to C AST
 3 pieces: global variable declarations, function definitions,
 other statements
*)
let rec stmt_sifter sifted = function
| [] -> sifted
| hd :: tl -> (match hd with
            | Sast.Vdecl(dt, id) ->
               stmt_sifter {s_globals = hd :: sifted.s_globals;
                                     s_main = sifted.s_main;
                                     s_funcs = sifted.s_funcs} tl
            | Sast.Assign(v, e, dt) ->
              stmt_sifter {s_globals = sifted.s_globals;
                          s_main = hd :: sifted.s_main;
                          s_funcs = sifted.s_funcs} tl
            | Sast.Expr(e) ->
              stmt_sifter {s_globals = sifted.s_globals;
```

```
                      s_main = hd :: sifted.s_main;
                      s_funcs = sifted.s_funcs} tl
            | Sast.NodeDef(id, e, dt) ->
                stmt_sifter {s_globals = sifted.s_globals;
                s_main = hd :: sifted.s_main;
                s_funcs = sifted.s_funcs} tl
            | Sast.AccessAssign(se1, se2, se3, dt) ->
                stmt_sifter {s_globals = sifted.s_globals;
                s_main = hd :: sifted.s_main;
                s_funcs = sifted.s_funcs} tl
            | Sast.GraphDef(id, el) ->
                stmt_sifter {s_globals = sifted.s_globals;
                s_main = hd :: sifted.s_main;
                s_funcs = sifted.s_funcs} tl
            | Sast.Return(e, dt) ->
                stmt_sifter {s_globals = sifted.s_globals;
                s_main = hd :: sifted.s_main;
                s_funcs = sifted.s_funcs} tl
            | Sast.Block(sl) ->
              let sifted_sl = stmt_sifter {s_globals = []; s_main = [];
                  s_funcs = []} sl in
              stmt_sifter {s_globals = sifted_sl.s_globals @ sifted.s_globals;
                      s_main = Block(sifted_sl.s_main) :: sifted.s_main;
                      s_funcs = sifted_sl.s_funcs @ sifted.s_funcs} tl
            | Sast.If(cond, s1, s2) ->
               let sifted_s1 = stmt_sifter {s_globals = []; s_main = [];
                   s_funcs = []} [s1] in
               let sifted_s2 = stmt_sifter {s_globals = []; s_main = [];
                   s_funcs = []} [s2] in
               let tmp = {s_globals = sifted_s1.s_globals @ sifted.s_globals;
                     s_main = sifted_s1.s_main @ sifted.s_main;
                     s_funcs = sifted_s1.s_funcs @ sifted.s_funcs} in
               stmt_sifter {s_globals = sifted_s2.s_globals @ tmp.s_globals;
                       s_main = sifted_s2.s_main @ tmp.s_main;
                       s_funcs = sifted_s2.s_funcs @ sifted.s_funcs} tl
            | Sast.For (tmp, iter, sl) ->
              let sifted_sl = stmt_sifter {s_globals = []; s_main = [];
                  s_funcs = []} sl in
              stmt_sifter {s_globals = sifted_sl.s_globals @ sifted.s_globals;
                      s_main = For(tmp, iter, sifted_sl.s_main) :: sifted.
                         s_main;
                      s_funcs = sifted_sl.s_funcs @ sifted.s_funcs} tl
            | Sast.While (cond, sl) ->
              let sifted_sl = stmt_sifter {s_globals = []; s_main = [];
                  s_funcs = []} sl in
              stmt_sifter {s_globals = sifted_sl.s_globals @ sifted.s_globals;
                      s_main = While(cond, sifted_sl.s_main) :: sifted.s_main
                         ;
                      s_funcs = sifted_sl.s_funcs @ sifted.s_funcs} tl
            | Sast.Fdecl(f) -> stmt_sifter {s_globals = sifted.s_globals;
                   s_main = sifted.s_main;
                   s_funcs = f :: sifted.s_funcs} tl
           )

```

```

let mappings = [("ine", Sast.Node); ("oute", Sast.Node); ("value", Sast.Node);
     ("nodes", Sast.Graph)]
let mem_vars = List.fold_left (fun m (k, v) -> StringMap.add k v m) StringMap.
    empty mappings

(* returns list of tuples mapping each elem of a list to consecutive
numbers starting from n and incrementing n by stride for each elem *)
let rec enum stride n = function
    [] -> []
  | hd::tl -> (n, hd) :: enum stride (n+stride) tl

  (* val string_map_pairs StringMap 'a -> (int * 'a) list -> StringMap 'a *)
  (* takes list of tuples (value, key) and adds them to the given map *)
let string_map_pairs map pairs =
   List.fold_left (fun m (i, n) -> StringMap.add n i m) map pairs

let find_max_index map =
   let bindings = StringMap.bindings map in
   let rec max cur = function
       | [] -> cur
       | hd :: tl -> if snd hd > cur then max (snd hd) tl else max cur tl
   in
   max 0 bindings


   (*
   returns the value associated with a given key,
   traversing through the list of maps until it finds the
   first occurrence of the key, or raises an error if none of
   the maps contain that key

   value: type
   key: variable name

   intended for things like: finding the type of a variable
   *)
let find_var var map_list =
   let rec finder var = function
       | m :: tl ->
             (try StringMap.find var !m
      with
      | Not_found -> finder var tl)
             | [] -> raise (Not_found )
   in
   finder var map_list

let str_to_type = function
   | "num" -> Sast.Num
   | "string" -> Sast.String
   | "bool" -> Sast.Bool
   | "graph" -> Sast.Graph
   | "node" -> Sast.Node
   | "dict" -> Sast.Dict(Sast.Void, Sast.Void)
```

```
    | "list" -> Sast.List(Sast.Void)
    | "void" -> Sast.Void
    | x -> raise (Failure ("unknown type: " ^ x))

(* for function args only, pass in a special type *)
let f_dt_to_type = function
    | Ast.Basic(dt) -> str_to_type dt
    | Ast.List(dt) -> Sast.List(str_to_type dt)
    | Ast.Dict(dtk,dtv) -> Sast.Dict(str_to_type dtk, str_to_type dtv)

(* converts a dataType to a string *)
let rec type_to_str = function
    | Sast.Num -> "num"
    | Sast.String -> "string"
    | Sast.Bool -> "bool"
    | Sast.Graph -> "graph"
    | Sast.Node -> "node"
    | Sast.Dict(dtk, dtv) -> "dict <" ^ type_to_str dtk ^ ", " ^ type_to_str
      dtv ^ ">"
    | Sast.List(dt) -> "list <" ^ type_to_str dt ^ ">"
    | Sast.Void -> "void"

let rec expr_type_str = function
    | Sast.NumLiteral(v, dt) -> "NumLiteral"
    | Sast.StrLiteral(v, dt) -> "StrLiteral"
    | Sast.ListLiteral(el, dt) -> "ListLiteral"
    | Sast.DictLiteral(kvl, dt) -> "DictLiteral"
    | Sast.Boolean(v, dt) -> "Boolean"
    | Sast.Id(v, dt) -> "Id"
    | Sast.Binop(e1, op, e2, dt) -> "Binop"
    | Sast.Call(v, el, dt) -> "Call"
    | Sast.Access(v, e, dt) -> "Access"
    | Sast.MemberCall(v, m, el, dt) -> "MemberCall"
    | Sast.Undir(v1, v2, dt) -> "Undir"
    | Sast.Dir(v1, v2, dt) -> "Dir"
    | Sast.UndirVal(v1, v2, w, dt) -> "UndirVal"
    | Sast.DirVal(v1, v2, w, dt) -> "DirVal"
    | Sast.BidirVal(w1, v1, v2, w2, dt) -> "BidirVal"
    | Sast.NoOp(v, dt) -> "NoOp"
    | Sast.Noexpr -> "Noexpr"
    (* returns the datatype of an Sast expressions *)
let get_sexpr_type = function
    | Sast.NumLiteral(v, dt) -> dt
    | Sast.StrLiteral(v, dt) -> dt
    | Sast.ListLiteral(el, dt) -> dt
    | Sast.DictLiteral(kvl, dt) -> dt
    | Sast.Boolean(v, dt) -> dt
    | Sast.Id(v, dt) -> dt
    | Sast.Binop(e1, op, e2, dt) -> dt
    | Sast.Call(v, el, dt) -> dt
    | Sast.Access(v, e, dt) -> dt
    | Sast.MemberCall(v, m, el, dt) -> dt
    | Sast.Undir(v1, v2, dt) -> Sast.Void
    | Sast.Dir(v1, v2, dt) -> Sast.Void
    | Sast.UndirVal(v1, v2, w, dt) -> Sast.Void
    | Sast.DirVal(v1, v2, w, dt) -> Sast.Void
    | Sast.BidirVal(w1, v1, v2, w2, dt) -> Sast.Void
    | Sast.NoOp(v, dt) -> Sast.Void
    | Sast.Noexpr -> Sast.Void

    (**********************)
    (* TRANSLATES AN SAST *)
    (**********************)
    (* determines whether a num string is an Int or a Float *)
let num_type num_str =
   let numregex = Str.regexp "-?[0-9]+$"
   in
   if Str.string_match numregex num_str 0 then Int else Float

let rec dt_to_ct = function
    | Sast.Num -> Float
    | Sast.String -> Cstring
    | Sast.Bool -> Int
    | Sast.Graph -> Graph (* TODO *)
    | Sast.Node -> Node (* TODO *)
    | Sast.List(dt) -> List(dt_to_ct dt) (* TODO *)
    | Sast.Dict(dtk, dtv) -> Ptr(Ptr(Entry)) (* TODO *)
    | Sast.Void -> Void

    (* the meat of the compiler *)
    (* actually converts Sast objects into strings of C code *)
let translate (env, sast_prg) =


(* Automatic Variables *)
(* certain translations require creating vars automatically
   keep track of all auto vars created so far, so that we
   don't repeat auto vars in C
*)

(* maps the given key to the next available int index
  returns the index/number that the key was mapped to

  Note: a key is a dots variable name
  ex. "key" : 3 := means that var "key" represents auto var "a3"
 *)
let create_auto env key dt =
     let ind = (find_max_index !(List.hd env.var_inds)+1) in
     let var_name = (match key with
        | "" -> "v" ^ string_of_int(ind)
        | _ -> key
     ) in
     (List.hd env.var_types) := StringMap.add var_name dt !(List.hd env.
        var_types); (* add type map *)
     (List.hd env.var_inds) := StringMap.add var_name ind !(List.hd env.
        var_inds); (* add index map *)
     ind
in
```

```

(*
 C equivalent:
  char str[50];
  int len;

  strcpy(str, "This is tutorialspoint.com");
  len = strlen(str);
*)
let string_len c_v =
 let cdt1 = Translate.get_cexpr_type c_v in
 if cdt1 = Cstring then
    let auto_var = "v" ^ string_of_int(create_auto env "" (Sast.Num)) in
     (auto_var, Block([
           Vdecl(Int, auto_var);
           Expr(Assign(Id(Int, auto_var),
                  Call(Int, "strlen", [c_v]))))]))
 else
   raise (Failure("only possible with string "))
 in

(* C requires special handling of string concatenation *)
let string_concat c_v1 c_v2 =
 let cdt2 = Translate.get_cexpr_type c_v2 in

 let len_c1 = ((string_len c_v1)) in
 let len_c2 = ((string_len c_v2)) in
 let len_new = Assoc(Binop(Int, Id(Int, (fst len_c1)), Add, Id(Int,(fst
    len_c2)))) in

 let auto_var = "v" ^ string_of_int(create_auto env "" (Sast.String)) in
   if cdt2 = Cstring then
  (auto_var,
   Block([
     (snd len_c1);
     (snd len_c2);
     Vdecl(Cstring, auto_var);
     Expr(Assign(
           Id(Cstring, auto_var),
           Call(Ptr(Void),
               "malloc",
              [Binop(Int, Call(Int, "sizeof", [Id(Void, "int")]), Mult, len_new
                 )])
          ));
     Expr(Call(Void,
         "strcpy",
         [Id(Cstring, auto_var);
          c_v2]));
     Expr(Call(Void,
         "strcat",
         [Id(Cstring, auto_var);
          c_v1]))
     ]))
 else
```

```
   raise (Failure("only accesible for strings"))

 in

let string_of_stmt c_v =
  let cdt = Translate.get_cexpr_type c_v in
  let s_dt = Translate.type_to_str cdt in
  let auto_var = "v" ^ string_of_int(create_auto env "" (Sast.String)) in
  (match cdt with
     | Int ->
        (auto_var, Block([
           Vdecl(Cstring, auto_var);
           Expr(Assign(Id(Cstring, auto_var),
                Call(Ptr(Void),
                    "malloc",
                   [Binop(Int,
                         Call(Int, "sizeof", [Id(Void, "char")]),
                         Mult,
                         Literal(Int, "400"))
                   ] )
               ));
           Call(Void,
               "itoa",
               [c_v;
                Id(Cstring, auto_var);
                Literal(Int,"10")
               ])
        ]))
     | Float ->
        (auto_var, Block([
               Vdecl(Cstring, auto_var);
                 Expr(Assign(Id(Cstring, auto_var),
                       Call(Ptr(Void),
                           "malloc",
                          [Binop(Int,
                                Call(Int, "sizeof", [Id(Void, "char")]),
                                Mult,
                                Literal(Int, "400"))
                          ] )
                      ));
                Expr(Call(Void,
                    "sprintf",
                    [Id(Void, auto_var);
                      Literal(Cstring,"%d.%02u");
                      Cast(Int, c_v);
                      Cast(Int,
                          (Binop(Float,
                               (Binop(Float, c_v, Sub, Cast(Int, c_v))),
                               Mult,
                               Literal(Int, "100"))
                          ))
                    ]))
                ]))
     | _ -> raise (Failure ("cannot convert type to cstring: " ^ s_dt)))
```

```
in


let rec build_args args = function
| [] -> args
| hd :: tl ->
   let arg_cstmts = translate_expr env hd in
   let arg_result = "v" ^ string_of_int (find_max_index !(List.hd env.var_inds
       )) in (* result of arg translation *)
   let arg_type = dt_to_ct (get_sexpr_type hd) in
   build_args ((arg_cstmts, Deref(arg_type, Id(Ptr(arg_type), arg_result))) ::
        args) tl
and

translate_expr env = function
   | Sast.NumLiteral(l, dt) ->
      let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in
      Block([ Vdecl(Ptr(Float), result_var);
                 Expr(Assign(Id(Ptr(Float), result_var),
                     Call(Ptr(Void), "malloc", [ Call(Int, "sizeof", [Id(Void, "
                        float")] ) ])
                     ));
               Expr(Assign( Deref(Float, Id( Ptr(Float), result_var)), Literal(
                   Float, l)))
           ])
   | Sast.Boolean(b, dt) ->
      let bool_val = if b = Ast.True then Literal(Int, "1") else Literal(Int,
          "0") in

      let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in
      Block([ Vdecl(Ptr(Int), result_var);
                 Expr(Assign(Id(Ptr(Int), result_var),
                     Call(Ptr(Void), "malloc", [ Call(Int, "sizeof", [Id(Void, "
                        int")] ) ])
                     ));
               Expr(Assign(Deref(Int, Id( Ptr(Int), result_var)),
                    bool_val
                   )
               )
           ])
   | Sast.StrLiteral(l, dt) ->
      let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in
         Block([
           (*create string*)
           Vdecl(Ptr(Cstring), result_var); (* char **result *)
           Expr(Assign(Id(Ptr(Cstring), result_var),
                     Call(Ptr(Void), "malloc", [ Call(Int, "sizeof", [Id(Void,
                        "char *")] ) ])
                     )
          ); (* *result = malloc(strlen(literal)) *)
          Expr(Assign(Deref(Cstring, Id(Ptr(Cstring), result_var)),
                    Call(Ptr(Void), "malloc",
                        [ Binop(Int, Call(Int, "strlen", [Literal(Cstring,
                        l)]), Add, Literal(Int, "1")) ]
                     )
                )
         ); (* strcpy( *result, literal) *)
         Expr(Call(Ptr(Void),
              "strcpy", [Deref(Cstring, Id(Ptr(Cstring), result_var));
                         Literal(Cstring, l)
                        ]
                 )
         )
       ])
   | Sast.ListLiteral(el, dt) ->
      let c_dt = dt_to_ct dt in
      let elem_stype = get_list_type dt in
      let elem_ctype = dt_to_ct elem_stype in
      let enq_func = (match elem_stype with
                  | Num -> "num_add_back"
                  | String -> "string_add_back"
                  | Node -> "node_add_back"
                  | Graph -> "graph_add_back"
                  | _ -> raise (Failure("can not enqueue this datatype"))
                 ) in
      let temp_list = "v" ^ string_of_int(create_auto env "" (dt)) in (* the
         variable containing the list *)

      let rec build_enqueue ops = function
      | [] -> ops
      | hd :: tl ->
         let elem_c = translate_expr env hd in (* translate list element being
             added *)
         let elem_result = "v" ^ string_of_int (find_max_index !(List.hd env.
            var_inds)) in (* get result of element translation *)
         let enq_call = Call(c_dt, enq_func, [Deref(c_dt, Id(Ptr(c_dt),
            temp_list));
                                  Id(Ptr(elem_ctype), elem_result) (*
                                     element translation result *)
                                 ]
                   ) in
         build_enqueue ((elem_c, enq_call) :: ops) tl
      in
      let enq_calls = build_enqueue [] el in (* get calls for each element in
         the list *)

      let rec build_list stmts = function
      | [] -> stmts
      | hd :: tl ->
         let en_stmt = Block([(fst hd);
                        Expr(Assign(Deref(c_dt, Id(Ptr(c_dt), temp_list)),
                                (snd hd)
                            )
                         )
                   ]) in
         build_list (en_stmt :: stmts) tl
      in
      let c_stmts = build_list [] enq_calls in (* combines the element
        translation and enqueue calls *)

    let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in (* will
         equal the temporary list created earlier *)

    Block(([Vdecl(Ptr(c_dt), temp_list);
            Expr(Assign(Id(Ptr(c_dt), temp_list),
                  Call(Ptr(Void), "malloc", [ Call(Int, "sizeof", [Id(Void,
                     "list_t *")]) ])
            ));
            Expr(Assign(Deref(c_dt, Id(Ptr(c_dt), temp_list)), Id(Void, "NULL"
               )));
            Vdecl(Ptr(c_dt), result_var)
           ]
          @
          (List.rev (Expr(Assign(Id(Ptr(c_dt), result_var), Id(Ptr(c_dt),
             temp_list)))
           :: (List.rev c_stmts)
          )))) (* add the assignment to the end of enqueue calls *)

  | Sast.DictLiteral(kvl, dt) ->
    DictLiteral(dt_to_ct dt,
          List.map (fun f -> (translate_expr env (fst f), translate_expr
             env (snd f))) kvl)(* TODO *)
  | Sast.Id(v, dt) ->
      let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in
      let index = "v" ^ string_of_int(find_var v env.var_inds) in (* see if
         id exists, get the num index of the var *)
      let v_type = dt_to_ct dt in
      Block([
            Vdecl(Ptr(v_type), result_var);
            Expr(Assign(Id(Ptr(v_type), result_var), Ref(Ptr(v_type), Id(
               v_type, index) )))
         ])
  | Sast.Binop(e1, op, e2, dt) ->
    let c_dt = dt_to_ct dt in

    let ce1 = translate_expr env e1 in
     let result_e1 = "v" ^ string_of_int (find_max_index !(List.hd env.
        var_inds)) in (* get result var of e1's translation *)
     let e1_cdt = dt_to_ct (get_sexpr_type e1) in (*gets is the c data type
         of the expression*)

    let ce2 = translate_expr env e2 in
     let result_e2 = "v" ^ string_of_int (find_max_index !(List.hd env.
        var_inds)) in (* get result var of e1's translation *)
     let e2_cdt = dt_to_ct (get_sexpr_type e2) in

    let cdt1 = Translate.get_cexpr_type ce1 in
    let cdt2 = Translate.get_cexpr_type ce2 in

    let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in (*
        create a new auto_var to store THIS EXPR'S result *)
    let result_decl = Vdecl(Ptr(c_dt), result_var) in (* declare this expr's
          result var *)

      let binop_func =
         (match op with
          | Add ->
            (match e1_cdt with
             | Float ->
                (match e2_cdt with
                 | Float -> Translate.Binop(Float,
                                   Deref(e1_cdt, Id(Ptr(e1_cdt), result_e1)
                                      ),
                                op, Deref(e2_cdt, Id(Ptr(e2_cdt), result_e2
                                   )))
                 | Cstring ->
                   let float_convert = string_of_stmt ce1 in
                   Block(
                   [(snd float_convert) ;
                    translate_expr env (Sast.Binop(Id((fst float_convert),
                       String), Add, e2, String))])
                 | _ -> raise(Failure("With the type checking in Sast, this
                   should never be reached..."))
                )
             | Cstring ->
                (match e2_cdt with
                 | Float ->
                   let float_convert = string_of_stmt ce2 in
                   Block([(snd float_convert) ;
                    translate_expr env (Sast.Binop(Id((fst float_convert),
                       String), Add, e1, String))])
                 | Cstring ->
                    let c_string = string_concat ce1 ce2 in
                    Block([(snd c_string)])
                 | Int ->
                   let int_convert = string_of_stmt ce2 in
                   Block([(snd int_convert) ;
                    translate_expr env (Sast.Binop(Id((fst int_convert),
                       String), Add, e1, String))])
                 | _ -> raise(Failure("With the type checking in Sast, this
                   should never be reached..."))
                )
             | Graph ->
                (match e2_cdt with
                 | Node ->
                   Call(Graph, "graph_plus_node", [Deref(Graph, Id(e1_cdt,
                      result_e1));
                                         Deref(Node, Id(e2_cdt, result_e2))
                                            ])
                 | Graph ->
                   Nostmt
                 | _ -> raise(Failure("With the type checking in Sast, this
                   should never be reached..."))
              )
             | Node ->
                (match e2_cdt with
                  | Node ->
                   Call(Graph, "node_plus_node", [Deref(Node, Id(e1_cdt,
                   result_e1));
                   Deref(Node, Id(e2_cdt, result_e2))])

                  | Graph -> Translate.Binop(cdt1, ce1, op, ce2) (*TODO*)
                  | _ -> raise(Failure("With the type checking in Sast, this
                     should never be reached..."))
                )
             | List(dt) -> Translate.Binop(cdt1, ce1, op, ce2) (*TODO*)
             | Int ->
                (match e2_cdt with
                  | Cstring ->
                     let int_convert = string_of_stmt ce1 in
                     Block(
                     [(snd int_convert) ;
                      translate_expr env (Sast.Binop(Id((fst int_convert),
                         String), Add, e2, String))])
                  | Int -> Translate.Binop(Int,
                                     Deref(e1_cdt, Id(Ptr(e1_cdt), result_e1)
                                         ),
                                  op, Deref(e2_cdt, Id(Ptr(e2_cdt), result_e2
                                      )))
                  | _ -> raise (Failure("invalid operation"))
                )
             | _ -> raise (Failure("Invalid c type for + binop " ^ (Translate
                .type_to_str cdt2)))
            )
          | Sub ->
            (match e1_cdt with
             | Float -> Translate.Binop(Float,
                                     Deref(e1_cdt, Id(Ptr(e1_cdt), result_e1)
                                         ),
                                  op, Deref(e2_cdt, Id(Ptr(e2_cdt), result_e2
                                      )))
             | Graph ->
                (match cdt2 with
                  | Node -> Translate.Binop(cdt1, ce1, op, ce2) (* TODO *)
                  | Graph -> Translate.Binop(cdt1, ce1, op, ce2) (* TODO *)
                  | _ -> raise(Failure("With the type checking in Sast, this
                     should never be reached..."))
                )
             | _ -> raise(Failure("With the type checking in Sast, this
                should never be reached..."))
            )
          | Mult | Div -> Translate.Binop(Float, ce1, op, ce2)
          | Equal | Neq ->
          (* This one isn't complete, dict maps to what c type? confusion *)
            (match e1_cdt with
            | Float -> Call(Int, "float_equals", [Id(e1_cdt, result_e1);
                                           Id(e2_cdt, result_e2)])
             | Int -> Translate.Binop(Int, Deref(e1_cdt, Id(Ptr(e1_cdt),
                result_e1)),
                                  op, Deref(e2_cdt, Id(Ptr(e2_cdt), result_e2
                                      )))
             | Cstring ->
                  (* (strcmp(check,input) = 0) *)
                  let auto_var = "v" ^ string_of_int(create_auto env "" (Sast.
                     Num)) in
                  Assign(Id(Int, auto_var), (Call(Int, "strcmp", [ce1;ce2])))
                      ;
             | Graph ->
                (match e2_cdt with
                  | Graph ->
                        (match op with
                        | Equal -> Call(Int, "graph_equals", [Deref(Ptr(Graph)
                            , Id(e1_cdt, result_e1)); Deref(Ptr(Graph), Id(
                            e2_cdt, result_e2))])
                        | Neq -> Translate.Binop(Int, Call(Int, "graph_equals"
                            ,
                                                  [Deref(Ptr(Graph), Id(e1_cdt
                                                      , result_e1));
                                                   Deref(Ptr(Graph), Id(e2_cdt
                                                      , result_e2))
                                                  ]),
                                            op, Literal(Int, "1"))
                        | _ -> raise(Failure("With the type checking in Sast,
                            this should never be reached..."))
                        )

                  | _ -> raise(Failure("With the type checking in Sast, this
                     should never be reached..."))
                )
             | Node ->
                (match e2_cdt with
                  | Node -> Translate.Binop(cdt1, ce1, op, ce2)
                  | _ -> raise(Failure("With the type checking in Sast, this
                     should never be reached..."))
                )
             | List(dt) -> Translate.Binop(cdt1, ce1, op, ce2) (*TODO*)
             | Void -> Translate.Binop(cdt1, ce1, op, ce2) (*TODO*)
             | _ -> raise (Failure("Invalid c type for ==/!= binop"))
            )
          | Less | Leq | Greater | Geq ->
            (match e1_cdt with
             | Float -> Translate.Binop(Float,
                                     Deref(e1_cdt, Id(Ptr(e1_cdt), result_e1)
                                         ),
                                  op, Deref(e2_cdt, Id(Ptr(e2_cdt), result_e2
                                      )))
             | Int ->
                Translate.Binop(Int,
                                     Deref(e1_cdt, Id(Ptr(e1_cdt), result_e1)
                                         ),
                                  op, Deref(e2_cdt, Id(Ptr(e2_cdt), result_e2
                                      )))
             | Long -> Translate.Binop(Long,
                                     Deref(e1_cdt, Id(Ptr(e1_cdt), result_e1)
                                         ),
                                 op, Deref(e2_cdt, Id(Ptr(e2_cdt), result_e2
                                    )))
             | Cstring ->
                 let auto_var = "v" ^ string_of_int(create_auto env "" (Sast.
                     Num)) in
                   Block([Vdecl(Int, auto_var);
                         Assign(Id(Int, auto_var), (Call(Int, "strcmp", [ce1;
                             ce2])))
                       ]);
             | _ -> raise(Failure("With the type checking in Sast, this
                 should never be reached..."))
           )
         | LogAnd | LogOr -> Translate.Binop(Int,ce1,op,ce2)
       ) in
       Block([
           ce1;
           ce2;
           result_decl;
           Expr(Assign(Id(Ptr(c_dt), result_var),
                     Call(Ptr(Void), "malloc", [ Call(Int, "sizeof",
                     [Id(Void, Translate.type_to_str c_dt)] ) ])
           ));
           Expr(Assign(Deref(c_dt, Id(Ptr(c_dt), result_var)), Assoc(
               binop_func)
           ))(* store the result of Access in our result_var *)
       ])
   | Sast.Call(func_name, el, dt) ->
       let return_type = dt_to_ct dt in
       (match func_name with
       | "print" ->
          let rec print_builder elems = function
          | [] -> List.rev elems
          | hd :: tl ->
             let hd_type = get_sexpr_type hd in
             let print_expr = translate_expr env hd in (* elem to print *)
             let print_result = "v" ^ string_of_int (find_max_index !(List.
                 hd env.var_inds)) in (* result of elem translation *)
             let deref_print_var = Deref(Node, Id(Ptr(Node), print_result))
                 in
             let print_type = dt_to_ct hd_type in (* type of elem *)

             (match hd_type with
              | Num | String | Bool ->
                 print_builder (Block([print_expr;
                                   Expr(Call(Void, "printf", [Literal(Cstring
                                       , get_fmt_str print_type);
                                                   Deref(print_type, Id(
                                                       Ptr(print_type),
                                                       print_result))
                                                   ]))
                                   ]) :: elems)
                              tl
              | Node ->
```

```
                    print_builder (Block([ print_expr;
                                    Expr(Call(Void, "printf", [Literal(
                                       Cstring, "%s"); Literal(Cstring, "N-"
                                       )] ));
                                    Expr(Call(Void, "printf", [Literal(
                                       Cstring, "%d"); Cast(Int,
                                       deref_print_var) ] ));
                                    Expr(Call(Void, "printf", [Literal(
                                       Cstring, "%s"); Literal(Cstring, "
                                       (\\\"")]));
                                    Expr(Call(Void, "printf", [Literal(
                                       Cstring, "%s"); Cast(Cstring, Member(
                                       Ptr(Void), deref_print_var, "data"))]
                                    ));
                                    Expr(Call(Void, "printf", [Literal(
                                       Cstring, "\\\")")]))
                                  ]) :: elems)
                             tl
                 | List(dt) ->
                    let print_loop = translate_stmt env
                                 (Sast.For("elem", hd,
                                        [Expr(Call("print", [Id("elem", dt)
                                           ], Sast.Void));
                                         Expr(Call("print", [StrLiteral(", "
                                            , String)], Sast.Void))
                                        ])) in

                   print_builder
                     (
                        (* b/c of building the list up backwards,
                          this list must be declared in reverse order

                           // c translation:
                           // print(num_list)
                           list_t* auto;
                           for (auto = num_list; auto; auto = auto->next) {
                             print( *auto );
                           }
                         *)
                       [

                        Expr(Call(Void, "printf", [Literal(Cstring, "]")]));
                        print_loop;
                        Expr(Call(Void, "printf", [Literal(Cstring, "[")]))


                       ]
                        @ elems
                     )
                     tl

                 | Dict(dtk, dtv) ->

                    (* build the print value statement for the specific key
                      type *)
                    (*
                     // C code:
                     int i;
                     entry_t *temp;
                     void *key;
                     /* print "{"; */
                     int first = 1;
                     for(i = 0; i < TABLE_SIZE; i = i + 1) {
                        for(temp = d[i]; temp; temp = temp->next) {
                           key = temp->key;
                           if(first) {
                              first = 0;
                              /* print key, ": ", value */
                           } else {
                              /* print ", " , key, ": ", value */
                           }
                        }
                     }
                    *)
                   let print_loop = translate_stmt env
                                  (Sast.For("$key", hd,
                                         [Expr(Call("print",
                                                [Id("$key", dtk)],
                                                Sast.Void));
                                          Expr(Call("print",
                                                [StrLiteral(": ", String)],
                                                Sast.Void));
                                          Expr(Call("print",
                                                [Sast.Access(hd, Id("$key",
                                                   dtk), dtv)],
                                                Sast.Void));
                                          Expr(Call("print",
                                                [StrLiteral(", ", String)],
                                                Sast.Void));
                                         ])) in

                   print_builder
                     (
                        (* b/c of building the list up backwards,
                          this list must be declared in reverse order

                           // c translation:
                           // print(num_list)
                           list_t* auto;
                           for (auto = num_list; auto; auto = auto->next) {
                            print( *auto );
                           }
                         *)
                      [

                       Expr(Call(Void, "printf", [Literal(Cstring, "}")]));
                       print_loop;
                       Expr(Call(Void, "printf", [Literal(Cstring, "{")]))


                     ]
                      @ elems
                    )
                    tl
               | Graph -> print_builder (Nostmt :: elems) tl (*TODO*)
               | Void -> raise (Failure "stop trying to print Void -- it's
                   not gonna happen")
              )
           in
           Block( print_builder [] el (* TODO *) )
        | "len" ->
        (*func_name, el, dt*)
              let elem_c = (translate_expr env (List.hd el)) in
              let arg_e = "v" ^ string_of_int (find_max_index !(List.hd env.
                 var_inds)) in
              let arg_dt = get_sexpr_type (List.hd el) in
              let arg_id = Id((dt_to_ct arg_dt), arg_e) in

              let result_var = "v" ^ string_of_int(create_auto env "" (dt))
                 in (* create a new auto_var to store THIS EXPR'S result *)
              let result_decl = Vdecl(Ptr(Float), result_var) in
              let final_result = Id(Ptr(Float), result_var) in
              (match arg_dt with
              | List(dt) ->
                    Block([
                       elem_c;
                       result_decl;
                       Expr(Assign(Id(Ptr(Float), result_var),
                          Call(Ptr(Void), "malloc", [ Call(Int, "sizeof", [Id
                             (Void, "float")] ) ])
                          )
                       );

                       Expr(Assign( Deref(Float, final_result),
                                 Cast((Float),
                                     Call(Int, "list_len", [ Deref((dt_to_ct
                                        arg_dt), arg_id) ]))
                       ));
                     ])
              | Dict(dtk, dtv) ->
                 Block([
                       elem_c;
                       result_decl;
                       Expr(Assign(Id(Ptr(Float), result_var),
                          Call(Ptr(Void), "malloc", [ Call(Int, "sizeof", [Id
                             (Void, "float")] ) ])
                          )
                       );

                       Expr(Assign( Deref(Float, final_result),
                                 Cast((Float),
                                     Call(Int, "dict_len", [ Deref((dt_to_ct
                                        arg_dt), arg_id) ]))
                       ));

                     ])
              | _ -> raise (Failure "len not implemented for this type")
              )
        | "min" | "max" ->
              let elem_c = (translate_expr env (List.hd el)) in
              let arg_e = "v" ^ string_of_int (find_max_index !(List.hd env.
                 var_inds)) in
              let arg_dt = get_sexpr_type (List.hd el) in
              let arg_id = Id((dt_to_ct arg_dt), arg_e) in

               (match arg_dt with
                 | List(dt) ->
                    let result_var = "v" ^ string_of_int(create_auto env "" (
                       dt)) in (* create a new auto_var to store THIS EXPR'S
                        result *)
                     let e_list_type = (get_list_type arg_dt) in
                     let result_decl = Vdecl(Ptr(Float), result_var) in
                     let final_result = Id(Ptr(Float), result_var) in

                      (match e_list_type with
                        | Num ->
                           let fname = "num_list_" ^ func_name in
                           Block([
                               elem_c;
                               result_decl;
                               Expr(Assign(Id(Ptr(Float), result_var),
                                  Call(Ptr(Void), "malloc", [ Call(Int, "
                                      sizeof", [Id(Void, "float")] ) ])
                                  )
                               );

                               Expr(Assign( Deref(Float, final_result),
                                          Cast((Float),
                                              Call(Float, fname, [ Deref((
                                                 dt_to_ct arg_dt), arg_id)
                                                 ]))
                               ));
                             ])
                        | _ -> raise (Failure ("cannot do min max ")))
                 | Dict(dtk, dtv) ->
                      (match dtv with
                        | Num ->
                            let fname = "num_dict_" ^ func_name in
                            let d_k = dt_to_ct dtk in
                            let result_var = "v" ^ string_of_int(create_auto
                                 env "" (dtk)) in (* create a new auto_var
                                to store THIS EXPR'S result *)
                            let result_decl = Vdecl(Ptr(d_k), result_var) in
                            let final_result = Id(Ptr(d_k), result_var) in

                             Block([
                               elem_c;
                               result_decl;
                               Expr(Assign(Id(Ptr(d_k), result_var),
                                  Call(Ptr(Void), "malloc", [ Call(Int, "
                                      sizeof", [Id(Void, Translate.
                                      type_to_str(d_k))] ) ])
                                  )
                               );

                               Expr(Assign(Deref(d_k, final_result),
                                        Cast((d_k),
                                        Call(Ptr(Void), fname, [ Deref((
                                            dt_to_ct arg_dt), arg_id) ]))
                               ));
                             ])
                        | _ -> raise (Failure ("cannot do min max ")))
                 | _ -> raise (Failure("can not enqueue this datatype")))
        | _ ->
           let c_args = List.rev (build_args [] el) in
           let c_stmts = List.map (fun t -> (fst t)) c_args in
           let result_params = List.map (fun t -> (snd t)) c_args in
           let func_index = "f" ^ string_of_int(find_var func_name env.
               func_inds) in
           let call_result = "v" ^ string_of_int(create_auto env "" (dt)) in
           Block(c_stmts
               @
               [
                Vdecl(Ptr(return_type), call_result);
                Expr(Assign(Id(Ptr(return_type), call_result),
                         Call(Ptr(Void), "malloc", [ Call(Int, "sizeof", [Id
                             (Void, Translate.type_to_str return_type)] ) ])
                     )
                 );
                 Expr(Assign(Deref(return_type, Id(Ptr(return_type),
                     call_result)),
                         Call(dt_to_ct dt, func_index, result_params)
                    )
                 )
               ]
           )
    )

 | Sast.Access(e1, e2, dt) ->
    let c_dt = dt_to_ct dt in
    let e1_dt = get_sexpr_type e1 in
    let e2_dt = get_sexpr_type e2 in

    let c_e1 = translate_expr env e1 in (* translate e1 to c *)
      let result_e1 = "v" ^ string_of_int (find_max_index !(List.hd env.
          var_inds)) in (* get result var of e1's translation *)
      let e1_deref = Deref(dt_to_ct e1_dt, Id(Ptr(dt_to_ct e1_dt), result_e1
          )) in

    let c_e2 = translate_expr env e2 in (* translate e2 to c *)
```

```
      let result_e2 = "v" ^ string_of_int (find_max_index !(List.hd env.
         var_inds)) in (* get result var of e2's translation *)
      let e2_deref = Deref(dt_to_ct e2_dt, Id(Ptr(dt_to_ct e2_dt), result_e2
         )) in

     let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in (*
        create a new auto_var to store THIS EXPR'S result *)
     let result_decl = Vdecl(Ptr(c_dt), result_var) in (* declare this expr's
         result var *)
     let args = [e1_deref; e2_deref] in (* specific to access function calls
        *)
     let access_call = (match e1_dt with
                | List(dt) ->
                    Call(Ptr(Void), "list_access", args)
                | Dict(dtk, dtv) ->
                    let c_dtk = dt_to_ct dtk in
                    (match c_dtk with
                    | Float -> Call(Ptr(Void), "get_num", args)
                    | Cstring -> Call(Ptr(Void), "get_string", args)
                    | Graph -> Call(Ptr(Void), "get_graph", args)
                    | Node -> Call(Ptr(Void), "get_node", args)
                    | _ -> raise(Failure("unsupported dict type"))
                    )
                | _ -> raise(Failure("unsupported access"))
              ) in (* evaluate an Access expression *)
     Block([
            c_e1;
            c_e2;
            result_decl;
            Assign(Id(Ptr(c_dt), result_var), Expr(Cast(Ptr(c_dt),
               access_call)))(* store the result of Access in our result_var
               *)
          ])
   | Sast.MemberCall(e, f, el, dt) ->
     let c_dt = dt_to_ct dt in

     let c_e = translate_expr env e in (* translate e1 to c *)
     let result_e = "v" ^ string_of_int (find_max_index !(List.hd env.
        var_inds)) in (* get result var of e1's translation *)
     let e_dt = get_sexpr_type e in
     let c_id = Id(dt_to_ct e_dt, result_e) in

        (match f with
            | "enqueue" | "push" ->
            let e_list_type = (get_list_type e_dt) in
              let suffix = (if f = "enqueue" then "back" else "front") in
              let func_name =
                (match e_list_type with
                | Num -> "num_add_" ^ suffix
                | String -> "string_add_" ^ suffix
                | Node -> "node_add_" ^ suffix
                | Graph -> "graph_add_" ^ suffix
                | _ -> raise (Failure("can not enqueue this datatype"))) in

```

```
                  let elem_c = (translate_expr env (List.hd el)) in
                  let arg_e = "v" ^ string_of_int (find_max_index !(List.hd env
                      .var_inds)) in
                  let arg_dt = get_cexpr_type elem_c in
                  let arg_id = Id(arg_dt, arg_e) in

                  let result_var = "v" ^ string_of_int(create_auto env "" (dt))
                       in (* create a new auto_var to store THIS EXPR'S result
                      *)
                  let result_decl = Vdecl(Ptr(dt_to_ct e_dt), result_var) in (*
                       declare this expr's result var *)
                       (*let e1_cdt = dt_to_ct (get_sexpr_type e1) in *)
                  let final_result = Id(dt_to_ct e_dt, result_var) in

                  Block([
                    c_e;
                    elem_c;
                    result_decl;
                    Expr(Assign(final_result,
                                 Call(Ptr(Void), "malloc", [ Call(Int, "sizeof", [Id
                                      (Void, Translate.type_to_str c_dt)] ) ])
                            )
                    );
                    Expr(Assign(Deref((dt_to_ct e_dt), c_id),
                            Call(c_dt, func_name,
                                         [Deref((dt_to_ct e_dt), c_id);
                                          Deref((arg_dt), arg_id)])
                    ))
                         (* store the result of Access in our result_var *)
                    ])
              | "dequeue" | "pop" ->
                  let result_var = "v" ^ string_of_int(create_auto env "" (dt))
                       in (* create a new auto_var to store THIS EXPR'S result
                      *)
                  let result_decl = Vdecl(Ptr(c_dt), result_var) in (* declare
                      this expr's result var *)

                       Block([

                            c_e;
                            result_decl;
                            Expr(Assign(Deref((dt_to_ct e_dt), c_id),
                                    Call(c_dt, "pop",
                                                   [Deref((dt_to_ct e_dt), c_id)])
                            ))
                         (* store the result of Access in our result_var *)
                    ])
              | "peek" ->
                let result_var = "v" ^ string_of_int(create_auto env "" (dt))
                     in (* create a new auto_var to store THIS EXPR'S result *)
                let result_decl = Vdecl(Ptr(c_dt), result_var) in (* declare
                     this expr's result var *)
                let final_result = Id(dt_to_ct e_dt, result_var) in

              Block([

                    c_e;
                    result_decl;
                    Expr(Assign(final_result,
                          Call(c_dt, "peek",
                                  [Deref((dt_to_ct e_dt), c_id)])
                    ))
                     (* store the result of Access in our result_var *)
                 ])

             | "oute" | "ine" ->
                let func_name =
                   (match f with
                    | "oute" -> "out"
                    | "ine" -> "in"
                    | _ -> raise (Failure ("unexpected out/in func name"))
                   ) in
                let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in (* create a new auto_var to store THIS EXPR'S result *)
                let result_decl = Vdecl(Ptr(Ptr(Ptr(Entry))), result_var) in (* declare this expr's result var *)
                let final_result = Id(Ptr(Entry), result_var) in
                Block([
                     c_e;
                     result_decl;

                      Expr(Assign(
                         final_result,
                         Ref(Ptr(Ptr(Ptr(Entry))), Member(Ptr(Entry), Deref((dt_to_ct e_dt), c_id), func_name))
                      ))
                 ])
             | "remove" ->
                (match e_dt with
                 | Dict(dtk, dtv) ->
                    let func_name =
                    (match dtk with
                    | Num -> "num_dict_remove"
                    | String -> "string_dict_remove"
                    | Node -> "node_dict_remove"
                    | Graph -> "graph_dict_remove"
                    | _ -> raise (Failure("can not enqueue this datatype"))) in

                    let key_c = (translate_expr env (List.hd el)) in
                    let key_result = "v" ^ string_of_int (find_max_index !(List.hd env.var_inds)) in
                    let key_dt = get_cexpr_type key_c in
                    let arg_id = Id(key_dt, key_result) in

                    let result_var = "v" ^ string_of_int(create_auto env "" (dt)) in (* create a new auto_var to store THIS EXPR'S
                         result *)
                    let result_decl = Vdecl(Ptr(Void), result_var) in (*
                      declare this expr's result var *)
                    Block([
                      c_e;
                      key_c;
                      result_decl;
                      Expr(Call(Void, func_name,
                               [Deref((dt_to_ct e_dt), c_id);
                               Deref(key_dt, arg_id)])
                      );
                         (* store the result of Access in our result_var *)
                      ])

                 | _ -> raise (Failure ("not a dict"))
                )


             | "val" ->
                let result_var = "v" ^ string_of_int(create_auto env "" (dt))
                    in (* create a new auto_var to store THIS EXPR'S result
                   *)
                Block([
                   c_e;
                   Vdecl(Ptr(Cstring), result_var);
                   Expr(Assign(
                         Id(Cstring, result_var),
                         Call(Ptr(Void),
                           "malloc",
                           [Call(Int, "sizeof", [Id(Void, "char *")]
                         )])
                          ));
                   Expr(Assign(
                         Deref(Cstring, Id(Ptr(Cstring),result_var)),
                         Member(Cstring, Deref((dt_to_ct e_dt), c_id), "data"
                         )))
                ])
              | _ -> raise (Failure("not enqueue")))
         | Sast.Undir(v1, v2, dt) ->
             let v1_index = "v" ^ string_of_int (find_var v1 env.var_inds) in
             let v2_index = "v" ^ string_of_int (find_var v2 env.var_inds) in
             Call(Void, "connect_undir", [Id(Ptr(Node), v1_index); Id(Ptr(Node),
             v2_index)])
         | Sast.Dir(v1, v2, dt) ->
             let v1_index = "v" ^ string_of_int (find_var v1 env.var_inds) in
             let v2_index = "v" ^ string_of_int (find_var v2 env.var_inds) in
             Call(Void, "connect_dir", [Id(Ptr(Node), v1_index); Id(Ptr(Node),
             v2_index)])
         | Sast.UndirVal(v1, v2, w, dt) ->
             let v1_index = "v" ^ string_of_int (find_var v1 env.var_inds) in
             let v2_index = "v" ^ string_of_int (find_var v2 env.var_inds) in
             let w_c = translate_expr env w in
             let w_result = "v" ^ string_of_int (find_max_index !(List.hd env.
             var_inds)) in
```

```
        let w_deref = Deref(dt_to_ct (get_sexpr_type w),
                        Id(Ptr(dt_to_ct (get_sexpr_type w)),
                               w_result)) in

        Block([
           w_c;
           Call(Void, "connect_undir_weighted ", [Id(Ptr(Node), v1_index); Id
               (Ptr(Node),
           v2_index); w_deref])
        ])
    | Sast.DirVal(v1, v2, w, dt) ->
        let v1_index = "v" ^ string_of_int (find_var v1 env.var_inds) in
        let v2_index = "v" ^ string_of_int (find_var v2 env.var_inds) in
        let w_c = translate_expr env w in
        let w_result = "v" ^ string_of_int (find_max_index !(List.hd env.
            var_inds)) in
        let w_deref = Deref(dt_to_ct (get_sexpr_type w), Id(Ptr(dt_to_ct
        (get_sexpr_type w)), w_result)) in

        Block([
           w_c;
           Call(Void, "connect_dir_weighted ", [Id(Ptr(Node), v1_index); Id(
              Ptr(Node),
           v2_index); w_deref])
        ])
    | Sast.BidirVal(w1, v1, v2, w2, dt) -> Nostmt (* TODO *)
    | Sast.NoOp(s, dt) -> Nostmt (* TODO *)
    | Sast.Noexpr -> (Nostmt)
and
translate_stmt env = function
    | Sast.Block(sl) ->
        let csl = List.map (translate_stmt env) sl in
        Block(csl)
    | Sast.Expr(e) -> Expr(translate_expr env e)
    | Sast.Vdecl(dt, id) ->
        (List.hd env.var_types) := StringMap.add id dt !(List.hd env.
            var_types); (* add type map *)
        (List.hd env.var_inds) := StringMap.add id (find_max_index !(List.hd
            env.var_inds)+1) !(List.hd env.var_inds); (* add index map *)
        let index = "v" ^ string_of_int(find_var id env.var_inds) in
        (match dt with
          | Num -> Vdecl(Float, index)
          | String -> Vdecl(Cstring, index)
          | Bool -> Vdecl(Int, index)
          | Graph -> Block([Vdecl(Graph, index);
                      ]) (* C: graph_t *g1 = init_graph(); *)
          | Node -> Block([Vdecl((Node), index);])(* C: node_t *x = init_node
              (""); *)
          | List(dt) -> Vdecl(List(dt_to_ct dt), index) (* C: list_t *x; *)
          | Dict(dtk, dtv) -> Vdecl(Ptr(Ptr(Entry)), index) (* TODO *)
          | Void -> raise (Failure ("should not be using Void as a datatype")
              )
        )
    | Sast.Assign(v, e, dt) ->
```

```
        let ce = translate_expr env e in
        let e_result = "v" ^ string_of_int (find_max_index !(List.hd env.
            var_inds)) in
        let cv = translate_expr env v in
        let v_result = "v" ^ string_of_int (find_max_index !(List.hd env.
            var_inds)) in
        let v_type = get_sexpr_type v in
        let var_type = dt_to_ct v_type in

           Block([
              ce; (* translation of assignment *)
              cv; (* translation of asignee *)
              Expr(Assign(Deref(var_type, Id(Ptr(var_type), v_result)), Deref
                  (var_type, Id(Ptr(var_type), e_result))))
            ])
    | Sast.AccessAssign(e1, e2, e3, dt) -> (* check access first *)
      let e1_dt = get_sexpr_type e1 in
      let e2_dt = get_sexpr_type e2 in
      let e3_dt = get_sexpr_type e3 in
      let c_e1 = translate_expr env e1 in
       let result_e1 = "v" ^ string_of_int (find_max_index !(List.hd env.
           var_inds)) in (* get result var of e1's translation *)
       let e1_deref = Deref(dt_to_ct e1_dt, Id(Ptr(dt_to_ct e1_dt), result_e1
           )) in
      let c_e2 = translate_expr env e2 in
       let result_e2 = "v" ^ string_of_int (find_max_index !(List.hd env.
           var_inds)) in (* get result var of e1's translation *)
       let e2_deref = Deref(dt_to_ct e2_dt, Id(Ptr(dt_to_ct e2_dt), result_e2
           )) in
      let c_e3 = translate_expr env e3 in
       let result_e3 = "v" ^ string_of_int (find_max_index !(List.hd env.
           var_inds)) in (* get result var of e1's translation *)
       let e3_deref = Deref(dt_to_ct e3_dt, Id(Ptr(dt_to_ct e3_dt), result_e3
           )) in
      let args = [e1_deref; e2_deref; Id(dt_to_ct e3_dt, result_e3)] in
      let call = (match e1_dt with
                  | List(dt) ->
                    if (e2_dt = Sast.Num) then
                      if (e3_dt = dt) then
                        let c_dt = dt_to_ct dt in
                        (match c_dt with
                          | Float -> Expr(Call(Ptr(Void), "num_index_insert",
                              args))
                          | Cstring -> Expr(Call(Ptr(Void), "string_index_insert
                              ", args))
                          | Graph -> Expr(Call(Ptr(Void), "graph_index_insert",
                              args))
                          | Node -> Expr(Call(Ptr(Void), "node_index_insert",
                              args))
                          | _ -> raise(Failure("unsupported list type"))
                        )
                      else
                        raise(Failure("accessassign: expr right of = is not
                            same type as list"))
```

```
 else
  raise(Failure("AccessAssign: assign expr on left is wrong for list"))
| Dict(dtk, dtv) ->
 if (e2_dt = dtk) then
  if (e3_dt = dtv) then
   let c_dtk = dt_to_ct dtk in
   let c_dtv = dt_to_ct dtv in
   (* try to get rid of problem with & *)
   let auto_var = "v" ^ string_of_int(create_auto env "" dtv) in
   (match c_dtk with
   | Float -> Block([Vdecl(c_dtv, auto_var);
         Expr(Assign(Id(c_dtv, auto_var), e3_deref));
         Expr(Assign(e1_deref,
               Call(Ptr(Void), "put_num",
                 [e1_deref;
                  e2_deref;
                  Cast(Ptr(Void),Ref(c_dtv, Id(c_dtv,auto_var)))
                 ])
            )
         )
        ])
   | Cstring -> Block([Vdecl(c_dtv,auto_var);
             Expr(Assign(Id(c_dtv, auto_var), e3_deref));
             Expr(Assign(e1_deref, Call(Ptr(Void), "put_string",
                          [e1_deref;
                           e2_deref;
                           Cast(Ptr(Void),Ref(c_dtv, Id(c_dtv, auto_var)))
                          ])
                     )
             )
         ])
   | Node -> let auto_var2 = "v" ^ string_of_int(create_auto env "" dtk) in
        Block([Vdecl(c_dtk,auto_var2);
         Expr(Assign(Id(c_dtk,auto_var2),
           Deref(dt_to_ct e2_dt,
           Id(Ptr(dt_to_ct e2_dt),
           result_e2))));
         Vdecl(c_dtv,auto_var);
         Expr(Assign(Id(c_dtv,auto_var), e3_deref));
         Expr(Assign(e1_deref,
               Call(Ptr(Void), "put_node",
                 [e1_deref;
                 Cast(Node, Id(c_dtk,auto_var2));
                   Cast(Ptr(Void),Ref(c_dtv, Id(c_dtv,auto_var)))]))
         )
```

```
                              ])
                          | _ -> raise(Failure("unsupported dict type"))
                          )
                        else
                          raise(Failure("accessassign: expr right of = is not
                              dict value type"))
                      else
                        raise(Failure("accessassign: assign expr on left for dict
                             is wrong"))
                   | _ -> raise(Failure("unsupported access"))
               ) in
      Block([
          c_e1;
          c_e2;
          c_e3;
          call;
      ])

  | Sast.Return(e, dt) ->
      let c_e = translate_expr env e in
      let e_result = "v" ^ string_of_int (find_max_index !(List.hd env.
          var_inds)) in

      Block([
        c_e;
        Translate.Return(Deref(dt_to_ct dt, Id(Ptr(dt_to_ct dt), e_result)))
      ])

  | Sast.NodeDef (id, s, dt) ->
      let index = "v" ^ string_of_int(find_var id env.var_inds) in
      let c_s = translate_expr env s in
        let result_s = "v" ^ string_of_int (find_max_index !(List.hd env.
            var_inds)) in
       (match s with
        | Sast.Noexpr ->
            Block([
              c_s;
               Expr(Assign(Id(Node, index), Call(Void, "init_node", [Literal(
                   Cstring, "")])));
            Expr(Assign(Member(Ptr(Void), Id(Void, index), "data"), Literal(
                Cstring,"")))])
        | _ ->
          Block([
            c_s;
            Expr(Assign(Id(Node, index), Call(Void, "init_node", [Literal(
                Cstring, "")])));
            Expr(Assign(Member(Ptr(Void), Id(Void, index), "data"), Deref(
                Cstring, Id(Ptr(Cstring), result_s)) ))
          ])
      )
  | Sast.GraphDef(id, sl) ->
      let index = "v" ^ string_of_int(find_var id env.var_inds) in
      let edge_ops = List.map (fun f -> Expr(translate_expr env f)) sl in

```

```
    let rec find_vars stmts = function
    | [] -> stmts
    | hd :: tl ->
       let calls =
          (match hd with
          | Undir(n1, n2, dt) | Dir(n1, n2, dt)->
            let n1_index = "v" ^ string_of_int(find_var n1 env.var_inds) in
            let n2_index = "v" ^ string_of_int(find_var n2 env.var_inds) in
            [
             Expr(Call(Void, "add_node", [Id(Graph, index); Id(Node,
                 n1_index)]));
             Expr(Call(Void, "add_node", [Id(Graph, index); Id(Node,
                 n2_index)]))
           ]
          | DirVal(n1, n2, w, dt) | UndirVal(n1, n2, w, dt) ->
            let n1_index = "v" ^ string_of_int(find_var n1 env.var_inds) in
            let n2_index = "v" ^ string_of_int(find_var n2 env.var_inds) in
            [
              Expr(Call(Void, "add_node", [Id(Graph, index); Id(Node,
                  n1_index)]));
              Expr(Call(Void, "add_node", [Id(Graph, index); Id(Node,
                  n2_index)]))
            ]
          | _ -> raise (Failure ("unexpected type"))
         )
       in
       find_vars ( calls @ stmts) tl
    in
    let node_add_calls = find_vars [] sl in
    Block( (Expr(Assign(Id(Graph, index), Call(Void, "init_graph", []))) ::
        edge_ops)
           @ node_add_calls
    )

  | Sast.While (cond, sl) ->
     (match cond with
      | Binop(e1, op, e2, dt) ->
        let c_e1 = translate_expr env e1 in
        let e1_cdt = dt_to_ct (get_sexpr_type e1) in
        let e1_result = "v" ^ string_of_int (find_max_index !(List.hd env.
            var_inds)) in

        let c_e2 = translate_expr env e2 in
        let e2_cdt = dt_to_ct (get_sexpr_type e2) in
        let e2_result = "v" ^ string_of_int (find_max_index !(List.hd env.
            var_inds)) in

        let c_stmts = List.map (translate_stmt env) sl in
        Block([c_e1; c_e2;
             While(Translate.Binop(Int,
                                Deref(e1_cdt, Id(Ptr(e1_cdt), e1_result)),
                                op,
                                Deref(e2_cdt, Id(Ptr(e2_cdt), e2_result))
                               ),
                    c_stmts)
          ])
       | _ -> raise (Failure ("not a while loop condition expression"))
      )

      (* convert body *)

  | Sast.If (cond, s1, s2) ->

      (match cond with
       | Binop(e1, op, e2, dt) ->
          let c_e1 = translate_expr env e1 in
          let e1_cdt = dt_to_ct (get_sexpr_type e1) in
          let e1_result = "v" ^ string_of_int (find_max_index !(List.hd env.
              var_inds)) in

          let c_e2 = translate_expr env e2 in
          let e2_cdt = dt_to_ct (get_sexpr_type e2) in
          let e2_result = "v" ^ string_of_int (find_max_index !(List.hd env.
              var_inds)) in

          let then_stmts = translate_stmt env s1 in
          let else_stmts = translate_stmt env s2 in
          Block([c_e1; c_e2;
                If(Translate.Binop(Int,
                                Deref(e1_cdt, Id(Ptr(e1_cdt), e1_result)),
                                op,
                                Deref(e2_cdt, Id(Ptr(e2_cdt), e2_result))
                               ),
                    [then_stmts],
                    [else_stmts])
          ])
       | _ -> raise (Failure ("not a while loop condition expression"))
      )
  | Sast.For (key, iter, sl) ->
         let iter_stype = get_sexpr_type iter in
         let iter_ctype = dt_to_ct iter_stype in
         let c_iter = translate_expr env iter in (* evaluate the iterable *)
         let iter_result = "v" ^ string_of_int (find_max_index !(List.hd env.
             var_inds)) in (* result of iterable *)

         let loop_var = "v" ^ string_of_int(create_auto env key (Sast.Void))
             in
         let key_var = "v" ^ string_of_int(create_auto env key iter_stype) in
             (* the temp var in "for x in iterable " *)
         let csl = List.map (translate_stmt env) sl in
         Block(
         c_iter ::
         [
             (match iter_stype with
              | List(dt) ->
                  let key_var_type = dt_to_ct dt in
                  let iter_deref = Deref(iter_ctype, Id(Ptr(iter_ctype),
                      iter_result)) in
                Block([Vdecl(key_var_type, key_var); (* *)
                     Vdecl(iter_ctype, loop_var);
                     For(Assign(Id(iter_ctype, loop_var), iter_deref),
                        Id(iter_ctype, loop_var),
                        Assign(Id(iter_ctype, loop_var), Member(iter_ctype,
                           Id(Void, loop_var), "next")),

                        Expr(Assign(Id(key_var_type, key_var),
                                 Deref(key_var_type, Cast(Ptr(key_var_type),
                                    Member(Ptr(Void), Id(Void,loop_var), "
                                    data")))
                            )
                        )
                        :: csl
                    )
                ])
             | Dict(dtk, dtv) ->
                   let iter_deref = Deref(Ptr(Entry), Id(Ptr(Ptr(Entry)),
                      iter_result)) in
                   let int_var = "v" ^ string_of_int(create_auto env "" (Sast
                      .Num)) in
                   let for_loop =
                      For(Assign(Id(Int, int_var), Literal(Int, "0")),
                         Binop(Int, Id(Int, int_var), Ast.Less, Id(Int, "
                            TABLE_SIZE")),
                         Assign(Id(Int, int_var),
                               Binop(Int, Id(Int, int_var), Ast.Add,
                               Literal(Int, "1"))),
                         [For(Assign(Id(Ptr(Entry), loop_var), Access(Entry,
                            Assoc(iter_deref), Id(Int, int_var))),
                            Id(Entry, loop_var),
                            Assign(Id(Ptr(Entry), loop_var), Member(Entry, Id
                               (Void, loop_var), "next")),
                            ( Expr(Assign(Id(Ptr(Void), key_var),
                                     Member(Ptr(Void), Id(Void, loop_var), "
                                        key")
                               )
                             )
                             :: List.map (translate_stmt env) sl
                            )
                         )]
                       ) in
                   Block([Vdecl(Int, int_var);
                        Vdecl(Ptr(Entry), loop_var);
                        Vdecl(Ptr(Void), key_var);
                        If(iter_deref, [for_loop], [])
                   ])
             | Node ->
                Let iter_deref = Deref(Node, Id(Ptr(Node), iter_result)) in
                Block([Vdecl(Ptr(Node), key_var);
                     Expr(Assign(Id(Ptr(Node), key_var), iter_deref))
                ] @ csl)
             | Graph ->
                Block([Vdecl(Node, key_var);
                              Vdecl(List(Node), loop_var);
                              For(Assign(Id(List(Node), loop_var),
                              Member(List(Node), Id(Void, "*" ^ iter_result), "
                                  nodes")),
                                Id(List(Node), loop_var),
                                Assign(Id(List(Node), loop_var),
                                Member(List(Node), Id(Void, loop_var), "next")),
                                Expr(Assign(Id(Node, key_var), Cast(Node, Member(
                                    Node,
                                Id(List(Node), loop_var), "data")))) :: csl
                                )
                             ])
               | _ -> raise (Failure("for loop iter is not iterable"))
             )
         ])

    | Sast.Fdecl (func) -> Nostmt

                 (* ***** TODO *********)
                 (*
                 { crtype = Translate.Void;
                 cfname = "empty";
                 cformals = [(Int, "argc"); (Ptr(Cstring), "argv")];
                 cbody = []}
               *)

and
translate_fdecl env func = (
    (* add the parameter names to the env *)
    let formals = func.s_formals in
    let rtype = func.s_rtype in

    (* add formal variables to local scope variable maps *)
    let map_builder fmls m = (List.map (fun f -> m := (StringMap.add (snd f) (
        fst f) !m); "") formals) in
    let types_map = ref StringMap.empty in
      ignore (map_builder formals types_map);
    let fml_inds = enum 1 (find_max_index !(List.hd env.var_inds)) (List.map (
        fun f -> (snd f)) formals) in
    let inds_map = ref (string_map_pairs StringMap.empty fml_inds) in

    let func_env = {
      var_inds = inds_map :: env.var_inds;  (* var names to indices ex. x -> 1
          so that we can just refer to it as v1 *)
      var_types = types_map :: env.var_types; (* maps a var name to its type ex
          . x -> num *)
      func_inds = ref StringMap.empty :: env.func_inds; (* func names to
          indices ex. x -> 1 so that we can just refer to it as f1 *)
      func_obj = ref StringMap.empty :: env.func_obj;
      return_type = rtype;             (* what should the return type be of the
          current scope *)
    } in

    {crtype = dt_to_ct func.s_rtype;
    cfname = "f" ^ string_of_int(find_var func.s_fname env.func_inds);
    cformals = List.map (fun f -> (dt_to_ct (fst f), "v" ^ string_of_int(
       find_var (snd f) func_env.var_inds))) func.s_formals;
    cbody = (List.map (fun f -> translate_stmt func_env f) func.s_body)}
)
in

(* convert the sast_prg to cast_prg *)
let global_vars = List.map (fun f -> translate_stmt env f) sast_prg.s_globals
   in
let main_func = {crtype = Translate.Int;
             cfname = "main";
             cformals = [(Int, "argc"); (Ptr(Cstring), "argv")];
             cbody = List.map (fun f -> translate_stmt env f) sast_prg.s_main}
in
let cfunc_list = List.map (fun f -> translate_fdecl env f) sast_prg.s_funcs in
{globals = global_vars; cfuncs = List.rev (main_func :: List.rev cfunc_list)}

let print_bindings m =
   let bindings = StringMap.bindings m in
   let rec printer = function
      | [] -> print_endline("")
      | (k, v)::tl -> print_endline(k ^ string_of_int(v)) ; printer tl
   in
   printer bindings
```

Listing 21: analyzer.ml

A.7 translate.ml

```
(* c AST is a library that handles c Asts pretty prints a c file *)
module StringMap = Map.Make(String)

type ctype = | Float | Int | Long | Cstring
          | Array of ctype
          | List of ctype
          | Graph
          | Node
          | Ptr of ctype (* pointer to a data type *)
          | Void
          | Entry

type cstmt =
| Literal of ctype * string
| DictLiteral of ctype * (cstmt * cstmt) list
| Id of ctype * string                (* ids arget_cexpr_typee ints ex. Id(2)
   -> v2 *)
| Binop of ctype * cstmt * Ast.op * cstmt
| Assign of cstmt * cstmt             (* ex. Assign(2, 5) -> v2 = 5 *)
| Call of ctype * string * cstmt list (* return type of the function, function
    name, arguments *) (* Call(3, [Literal(5), Id(3)]) -> f3(5, v3) *)
| Access of ctype * cstmt * cstmt    (* array access: id[cexpr] *)
| Member of ctype * cstmt * string   (* id, member *)
| Cast of ctype * cstmt               (* ex. Cast(Int, Id(f1)) -> (int)(f1) *)
| Deref of ctype * cstmt                        (* ex. *var *)
| Ref of ctype * cstmt                          (* ex. &var *)
| Block of cstmt list
| Expr of cstmt
| Vdecl of ctype * string             (* (type, id) ex. Vdecl(Int, 2) -> int v2
   ; *)
| Return of cstmt
| If of cstmt * cstmt list * cstmt list
| For of cstmt * cstmt * cstmt * cstmt list (* assign, condition, incr, body
   -> ex. for (v1 = 3, v1 < 10; v1 = v1 + 1 *)
| While of cstmt * cstmt list
| Assoc of cstmt (* wrap the expression in parentheses *)
| Nostmt

type c_func = { crtype : ctype; (* c return type *)
            cfname : string; (* function name *)
            cformals : (ctype * string) list; (* (data type, id) list *)
            cbody : cstmt list;
           }

type cprogram = {
               globals : cstmt list; (* global variables -- Note: should ONLY
                  be Vdecl list *)
               cfuncs : c_func list;
            }

let rec type_to_str = function
| Float -> "float"
| Int -> "int"
| Long -> "long"
| Cstring -> "char *"
| Array(dt) -> type_to_str dt ^ "[]"
| List(dt) -> "list_t *"
| Node -> "node_t *"
| Graph -> "graph_t *"
| Ptr(dt) -> type_to_str dt ^ "*"
| Void -> "void"
| Entry -> "entry_t"

let fmt_str = function
| Float -> "%f"
| Int -> "%d"
| Cstring -> "%s"
| _ -> raise (Failure ("can't print other types directly"))

let rec get_cexpr_type = function
| Literal(dt, str) -> dt
(* | ListLiteral(dt, el) -> dt *)
| DictLiteral(dt, tl) -> dt
| Id(dt, id) -> dt
| Binop(dt, e1, op, e2) -> dt
| Assign(id, e1) -> Void
| Call(dt, id, el) -> dt
| Access(dt, id, e) -> dt
| Member(dt, stmt, m) -> dt
| Cast(dt, e) -> dt
| Ref(dt, e) -> dt
| Deref(dt, e) -> dt
| Assoc(e) -> get_cexpr_type e
| _ -> Void

let rec stmt_type_to_str = function
| Literal(dt, str) -> "Literal<" ^ type_to_str dt ^ ">"
| DictLiteral(dt, tl) -> "DictLiteral<" ^ type_to_str dt ^ ">"
| Id(dt, id) -> "Id<" ^ type_to_str dt ^ ">"
| Binop(dt, e1, op, e2) -> "Binop<" ^ type_to_str dt ^ ">"
| Assign(id, e1) -> "Assign<" ^ stmt_type_to_str id ^ ">"
| Call(dt, id, el) -> "Call<" ^ type_to_str dt ^ ">"
| Access(dt, id, e) -> "Access<" ^ type_to_str dt ^ ">"
| Member(dt, stmt, m) -> "Member<" ^ type_to_str dt ^ ">"
| Cast(dt, e) -> "Cast<" ^ type_to_str dt ^ ">"
| Ref(dt, e) -> "Ref<" ^ type_to_str dt ^ ">"
| Deref(dt, e) -> "Deref<" ^ type_to_str dt ^ ">"
| Block(sl) -> "Block"
| Expr(e) -> "Expr:" ^ stmt_type_to_str e
| Vdecl(dt, id) -> "Vdecl<" ^ type_to_str dt ^ ">"
| Return(e) -> "Return:" ^ stmt_type_to_str e
| If(cond, el1, el2) -> "If-then-Else"
| For(assign, cond, incr, sl) -> "For"
| While(cond, sl) -> "While"
| Assoc(e) -> stmt_type_to_str e
| Nostmt -> "Nostmt"

let op_to_str = function
| Ast.Add -> "+"
| Ast.Sub -> "-"
| Ast.Mult -> "*"
| Ast.Div -> "/"
| Ast.Equal -> "=="
| Ast.Neq -> "!="
| Ast.Less -> "<"
| Ast.Leq -> "<="
| Ast.Greater -> ">"
| Ast.Geq -> ">="
| Ast.LogAnd -> "&&"
| Ast.LogOr -> "||"

(* takes a c datatype and returns the print format string *)
let get_fmt_str = function
| Float -> "%.3f"
| Int -> "%d"
| Long -> "%l"
| Cstring -> "%s"
| Node -> raise (Failure "type node can't be directly printed")
| Entry | Graph -> raise (Failure "type requires iterable print handling")
| List(dt) | Array(dt) -> raise (Failure "type requires iterable print
   handling")
| Void -> raise (Failure ("can't directly print Void"))
| Ptr(dt) -> raise (Failure "can't directly print pointer")

let cvar_cnt = ref 0

let rec translate_stmt = function
| Literal(dt, v) ->
  (match dt with
   | Float -> v
   | Int -> v
   | Cstring -> "\"" ^ v ^ "\""
   | Array(adt) -> v
   | Void -> if v = "NULL" then v else raise (Failure "Void lit should only be
      'NULL'")
   | _ -> raise (Failure "invalid C literal type")
  )
| DictLiteral(dt, el) -> translate_stmt (Literal(Cstring, "TODO: dict literal"
   ))
| Id(dt, id) -> id
| Binop(dt, e1, op, e2) ->
   (* check if either e1 is a string or e2 is a string:
     different operation: concatenation
   *)
  translate_stmt e1 ^ " " ^ op_to_str(op) ^ " " ^ translate_stmt e2
| Assign(target, e) -> (translate_stmt target) ^ " = " ^ translate_stmt e
| Call(dt, id, el) ->
   id ^ "(" ^ (String.concat ", " (List.map translate_stmt el)) ^ ")"

| Access(dt, id, e) -> (translate_stmt id) ^ "[" ^ (translate_stmt e) ^ "]"
| Member(dt, id, m) -> (translate_stmt (Assoc(id))) ^ "->" ^ m
| Cast(dt, e) -> "(" ^ type_to_str dt ^ ")(" ^ translate_stmt e ^ ")"
| Ref(dt, e) -> "&(" ^ translate_stmt e ^ ")"
| Deref(dt, e) -> "*(" ^ translate_stmt e ^ ")"
| Block(sl) -> String.concat "\n" (List.map translate_stmt sl)
| Expr(e) -> translate_stmt e ^ ";"
| Vdecl(dt, id) ->
   (match dt with
    | Ptr(Ptr(Entry)) -> type_to_str dt ^ " " ^ id ^ " = NULL;"
    | List(vdt) -> type_to_str dt ^ " " ^ id ^ " = NULL;"
    | Graph -> type_to_str dt ^ " " ^ id ^ " = NULL;"
    | _ -> type_to_str dt ^ " " ^ id ^ ";"
   )

| Return(e) -> "return " ^ translate_stmt e ^ ";"
| If(cond, sl1, sl2) -> "if (" ^ translate_stmt cond ^ ") {\n" ^
   String.concat "\n" (List.map translate_stmt sl1) ^
   "\n} else {\n" ^
   String.concat "\n" (List.map translate_stmt sl2) ^
   "\n}"
| For(init, cond, incr, sl) -> "for (" ^ translate_stmt init ^ "; " ^
   translate_stmt cond ^ "; " ^
   translate_stmt incr ^ ") {\n" ^
   String.concat "\n" (List.map translate_stmt sl) ^
   "\n}"
| While(cond, sl) -> "while (" ^ translate_stmt cond ^ ") {\n" ^
```

```
   String.concat "\n" (List.map translate_stmt sl) ^
   "\n}"
| Assoc(e) -> "(" ^ (translate_stmt e) ^ ")"
| Nostmt -> ""


let translate_func func =
   (type_to_str func.crtype) ^ " " ^ func.cfname ^ " (" ^
   String.concat ", " (List.map (fun f -> (type_to_str (fst f)) ^ " " ^ snd f)
        func.cformals) ^
   ")\n{\n" ^
   String.concat "\n" (List.map translate_stmt func.cbody) ^
   "\n}\n"

(* eventually won't be used by analyzer.ml *)
let string_of_cfunc func =
   (type_to_str func.crtype) ^ " " ^ func.cfname ^ " (" ^
   String.concat ", " (List.map (fun f -> (type_to_str (fst f)) ^ " " ^ snd f)
        func.cformals) ^
    ")\n{\n" ^
   (String.concat "\n" (List.map translate_stmt func.cbody)) ^
   "\n}\n"

let translate_c (globals, cfuncs) =
   (* "\"graph.h\"" *)
   let libs = ["<stdio.h>"; "<stdlib.h>"; "<string.h>";
             "<dict.h>"]
   in

   (* now we are going to translate a program *)
   (String.concat "\n" (List.map (fun f -> "#include " ^ f) libs)) ^
   "\n" ^
   (String.concat "\n" (List.map translate_stmt globals)) ^
   "\n" ^
   (String.concat "\n" (List.map translate_func cfuncs))
```

Listing 22: translate.ml

A.8 compile.ml

```
(* ties together parsing scanning ast sast the whole fucking thing *)
open Ast
open Sast
open TypeConverter
open Translate
open Analyzer

(* translate version *)
let _ =
  let lexbuf = Lexing.from_channel stdin in
  let ast_prg =
    (try
       (Parser.program Scanner.token lexbuf) (* outputs the Ast from parsing
           and scanning *)
```

```
   with exn ->
        let curr = lexbuf.Lexing.lex_curr_p in
        let line = curr.Lexing.pos_lnum in
        let cnum = curr.Lexing.pos_cnum - curr.Lexing.pos_bol in
        let tok = Lexing.lexeme lexbuf in
        raise (Failure ("Parsing error: line " ^ string_of_int(line) ^ ", char
             " ^ string_of_int(cnum) ^ ", token " ^ tok))
   )
   in

     let print_decl = {
        s_fname = "print";
        s_rtype = Sast.Void;
        s_formals = [];
        s_body = [];
     } in
     let range_decl = {
        s_fname = "range";
        s_rtype = Sast.List(Sast.Num);
        s_formals = [];
        s_body = [];
     } in

(* set up default environment *)
 let sast_env =
    (* built-in function set-up *)
    let bf_names = [ "print"; "range";] in
    let bf_inds = enum 1 1 bf_names in
    let bf_ind_map = ref (string_map_pairs StringMap.empty bf_inds) in
    let bf_fdecl_map = ref (string_map_pairs StringMap.empty [(print_decl, "
        print"); (range_decl, "range")]) in
   (* build default symbol tables: *)
   {var_types = [ref StringMap.empty];
                 var_inds = [ref StringMap.empty];
                 func_obj = [bf_fdecl_map];
                 func_inds = [bf_ind_map];
                 return_type = Sast.Void} in

 (* convert Ast to Sast *)
 let sast_prg = convert_ast { cmds = List.rev ast_prg.cmds} sast_env in

 (* massage Sast into a form more suitable for C Ast *)
 (* i.e. split up variable declarations, function definitions, and other *)
 let sifted_prg = stmt_sifter {s_globals = []; s_main = []; s_funcs = []}
     sast_prg.s_cmds in

 (* construct Sast program object from sifted *)
 let sifted_prg = {s_globals = List.rev sifted_prg.s_globals;
                s_main = List.rev sifted_prg.s_main;
                s_funcs = List.rev sifted_prg.s_funcs} in

 (* set up default environ *)
 let trans_env =
    let bf_names = [ "print"; "range"; "len"; "min"; "max"] in
    let bf_inds = enum 1 1 bf_names in
    let bf_ind_map = ref (string_map_pairs StringMap.empty bf_inds) in
    let bf_fdecl_map = ref (string_map_pairs StringMap.empty [(print_decl, "
        print"); (range_decl, "range")]) in
   {var_types = [ref StringMap.empty];
                 var_inds = [ref StringMap.empty];
                 func_obj = [bf_fdecl_map];
                 func_inds = [bf_ind_map];
                 return_type = Sast.Void} in

 (* add declared functions to symbol tables *)
 let rec func_def_adder env = function
 | [] -> ignore()
 | hd :: tl ->
    (List.hd env.func_obj) := StringMap.add hd.s_fname hd !(List.hd env.
        func_obj);
    (List.hd env.func_inds) := StringMap.add hd.s_fname (find_max_index !(
        List.hd env.func_inds)+1) !(List.hd env.func_inds); (* add index map
        *)
    ignore(func_def_adder env tl)
 in
 ignore(func_def_adder trans_env sifted_prg.s_funcs);

(* convert Sast to C Ast *)
let cprg = translate(trans_env, sifted_prg) in

 (* output C code from C Ast *)
 print_endline (translate_c(cprg.globals, cprg.cfuncs))
```

Listing 23: compile.ml

B C Library Files

B.1 node.h

```
struct node;
typedef struct node node_t;
typedef struct entry entry_t;

struct node {
   char *data;
   entry_t **in;
   entry_t **out;
};

struct entry {
   void *key;
   void *value;
   struct entry *next;
};

/* initialize a new node that contains *data */
node_t *init_node(char *data);
```

```c

/* compares node data */
int node_compare(node_t *a, node_t *b, int (* comp) (void *a, void *b));

/* create undirected edge of weight 0 */
void connect_undir(node_t *a, node_t *b);

/* create weighted undirected edge */
void connect_undir_weighted(node_t *a, node_t *b, float weight);

/* create directed edge of weight 0 */
void connect_dir(node_t *src, node_t *dst);

/* create weighted directed edge */
void connect_dir_weighted(node_t *src, node_t *dst, float weight);

/* remove directed edge from src to dst */
void remove_dir_edge(node_t *src, node_t *dst);

/* remove undirected edge between a and b */
void remove_undir_edge(node_t *a, node_t *b);

/* deallocate node */
void free_node(node_t *n);
```
Listing 24: node.h

B.2 node.c

```c
#include <stdlib.h>
#include <stdio.h>
#include "dict.h"

/* initialize a new node that contains *data */
node_t *init_node(char *data) {
   node_t *n = (node_t *) malloc(sizeof(node_t));
   n->data = data;
   n->in = init_dict();
   n->out = init_dict();
   return n;
}

void free_node(node_t *n) {
   printf("freeing node at %x\n", (int) n);
   free(n->data);
   free(n);
}

/* TODO decide on default behavior */
/* compares node data, should return 0 if a == b, 1 if a > b, and -1 if a < b
   */
int node_compare(node_t *a, node_t *b, int (* comp)(void *a, void *b)) {
   /*
   if(comp == NULL)
```

```
        return *(a->data) == *(b->data);
        */
    return comp(a->data, b->data);
}

/* create undirected edge of weight 0 */
void connect_undir(node_t *a, node_t *b) {
    connect_dir(a, b);
    connect_dir(b, a);
}

/* create weighted undirected edge */
void connect_undir_weighted(node_t *a, node_t *b, float weight) {
    connect_dir_weighted(a, b, weight);
    connect_dir_weighted(b, a, weight);
}

/* create directed edge of weight 0 */
void connect_dir(node_t *src, node_t *dst) {
    connect_dir_weighted(src, dst, 0);
}

/* create weighted directed edge */
void connect_dir_weighted(node_t *src, node_t *dst, float weight) {

    void *a = malloc(sizeof(float));
    *(float *)a = weight;
    /* add dst to src->out */
    put_node(src->out, dst, a);

    /* add src to dst->in */
    put_node(dst->in, src, a);

    /*
    edgelist_t *e = (edgelist_t *) malloc(sizeof(edgelist_t));
    e->node = dst;
    e->weight = weight;
    e->previous = NULL;
    e->next = src->out;
    if(src->out != NULL)
        src->out->previous = e;
    src->out = e;

     add src to dst->in
    edgelist_t *f = (edgelist_t *) malloc(sizeof(edgelist_t));
    f->node = src;
    f->weight = weight;
    f->previous = NULL;
    f->next = dst->in;
    if(dst->in != NULL)
        dst->in->previous = f;
    dst->in = f;
    */
}
```

```c

/* remove directed edge from src to dst */
void remove_dir_edge(node_t *src, node_t *dst) {
   /*
   edgelist_t *e = src->out;
   while(e && e->node != dst)
      e = e->next;
   if(!e) {
      printf("there is no edge from %s to %s\n", (char *) src->data, (char *)
         dst->data);
      return;
   }

   edgelist_t *f;
   f = e->node->in;
   while(f && f->node != src)
      f = f->next;
   if(!f)
      printf("f is NULL\n");
   else {
      if(f->previous)
         f->previous->next = f->next;
      if(f->next)
         f->next->previous = f->previous;
      if(!f->next && !f->previous) {
         e->node->in = 0;
      }
      free(f);
   }
   if(e->previous)
      e->previous->next = e->next;
   if(e->next)
      e->next->previous = e->previous;
   if(!(e->next || e->previous))
      src->out = 0;
   free(e);
   e = NULL;
   */
}

/* remove undirected edge between a and b */
void remove_undir_edge(node_t *a, node_t *b) {
   remove_dir_edge(a, b);
   remove_dir_edge(b, a);
}

char * value(node_t *n) {
   return (n->data);
}
```
Listing 25: node.c

B.3 graph.h

```c
#include "node.h"
```

```c

typedef struct list list_t;
struct list {
   struct list *next;
   struct list *previous;
   void *data;
};

typedef struct graph graph_t;
struct graph {
   list_t *nodes;
   int count;
};

graph_t *init_graph();

void free_graph(graph_t * graph);

int contains(graph_t *graph, void *data, int (* comp)(void *a, void *b));

graph_t *add_node(graph_t *graph, const node_t *node);

int remove_node(graph_t *graph, node_t *node);

graph_t *plus(const graph_t *a, const graph_t *b);

graph_t *plus_equals(graph_t *a, const graph_t *b);

graph_t *minus(const graph_t *left, const graph_t *right);

graph_t *graph_copy(const graph_t *src);

int graph_equals(const graph_t *a, const graph_t *b);

graph_t *graph_plus_node(const graph_t *g, const node_t *n);

graph_t *node_plus_node(const node_t *n1, const node_t *n2);
```

Listing 26: graph.h

B.4 graph.c

```c
#include <stdlib.h>
#include <stdio.h>
#include "graph.h"

/* initialize empty graph */
graph_t *init_graph() {
   graph_t *g = (graph_t *) malloc(sizeof(graph_t));
   g->nodes = NULL;
   g->count = 0;
   return g;
}

```

```
/* deallocate graph */
void free_graph(graph_t * g) {
   if(g == NULL)
      return;
   list_t *temp = g->nodes;
   while(temp->next) {
      temp = temp->next;
      free(temp->previous);
   }
   free(temp);
   free(g);
}

/* check if graph contains data */
int contains(graph_t *g, void *data, int (* comp)(void *a, void *b)) {
   list_t *temp = g->nodes;
   while(temp)
      if(comp(((node_t *) temp->data)->data, data))
         return 1;
   return 0;
}

/* add a node to g by iterating through the list, returning if the node is
    found, and if not, adding it to the end */
graph_t *add_node(graph_t *g, const node_t *node) {
   if(g) {
      list_t *n = (list_t *)malloc(sizeof(list_t));
      n->data = (void *) node;
      list_t *temp = g->nodes;
      /* make temp point to last list_t in g->nodes */
      if(temp) {
         while(temp->next) {
            if(temp->data == node)
               return g;
            temp = temp->next;
         }
         temp->next = n;
      } else {
         g->nodes = n;
      }
      n->previous = temp;
      n->next = NULL;
      g->count++;
   } else {
      g = init_graph();
      add_node(g, node);
   }
   return g;
}


/* returns 0 on success, 1 if node not found */
int remove_node(graph_t *g, node_t *n) {
   if(n == NULL)
```

```
      return 1;
   list_t *temp = g->nodes;
   while(temp && temp->data != n)
      temp = temp->next;
   if(temp == NULL)
      return 1;
   if(temp->previous)
      temp->previous->next = temp->next;
   else
      g->nodes = temp->next;
   if(temp->next)
      temp->next->previous = temp->previous;
   if(!temp->next && !temp->previous) {
      g->nodes = 0;
   }
   free(temp);
   g->count--;
   return 0;
}

/* returns a graph containing all nodes in *a and all nodes in *b */
graph_t *plus(const graph_t *a, const graph_t *b) {
   graph_t *g = (graph_t *) malloc(sizeof(graph_t));
   g->nodes = NULL;
   g->count = 0;
   plus_equals(g, a);
   plus_equals(g, b);
   return g;
}

/* adds all nodes from *b to *a. returns a */
graph_t *plus_equals(graph_t *a, const graph_t *b) {
   list_t *temp;
   for(temp = b->nodes; temp; temp = temp->next)
      add_node(a, temp->data);
   return a;
}

/* removes all nodes of *right that exist in *left from *left */
graph_t *minus(const graph_t *left, const graph_t *right) {
   list_t *temp;
   graph_t *copy = graph_copy(left);
   for(temp = right->nodes; temp; temp = temp->next) {
      printf("removing temp = %x\n", (int) temp);
      int i = remove_node(copy, temp->data);
      if(i)
         printf("not removed!\n");
   }
   return copy;
}

graph_t *graph_copy(const graph_t *src) {
   graph_t *g = init_graph();
   list_t *temp;
```

```c
    for(temp = src->nodes; temp; temp = temp->next)
        add_node(g, temp->data);
    return g;
}

int graph_equals(const graph_t *a, const graph_t *b) {
    if(a == b)
        return 1;
    const list_t *temp_a = a->nodes;
    const list_t *temp_b = b->nodes;
    while(temp_a && temp_b) {
        if(temp_a->data != temp_b->data)
            return 0;
        temp_a = temp_a->next;
        temp_b = temp_b->next;
    }
    if(temp_a || temp_b)
        return 0;
    return 1;
}

graph_t *graph_plus_node(const graph_t *g, const node_t *n) {
    graph_t *copy;
    if(g)
        copy = graph_copy(g);
    else
        copy = init_graph();
    add_node(copy, n);
    return copy;

}

graph_t *node_plus_node(const node_t *n1, const node_t *n2) {
    graph_t *ret = init_graph();
    add_node(ret, n1);
    add_node(ret, n2);
    return ret;
}
```

Listing 27: graph.c

B.5 list.h

```c
 #include "graph.h"

/* copy constructors */
list_t *string_list_copy(const list_t *src);

list_t *num_list_copy(const list_t *src);

list_t *graph_list_copy(const list_t *src);

list_t *node_list_copy(const list_t *src);

```

```
/*
list_t *dict_copy(list_t *src);

list_t *other_copy(list_t *src);
*/

/* insertion */
list_t *string_add_front(list_t *l, char *data);

list_t *num_add_front(list_t *l, float *data);

list_t *graph_add_front(list_t *l, graph_t *data);

list_t *node_add_front(list_t *l, node_t *data);

/*
list_t *dict_add_front(list_t *l, entry_t **data);

list_t *other_add_front(list_t *l, void *data);

*/
list_t *string_add_back(list_t *l, char *data);

list_t *num_add_back(list_t *l, float *data);

list_t *graph_add_back(list_t *l, graph_t *data);

list_t *node_add_back(list_t *l, node_t *data);

/*
list_t *dict_add_back(list_t *l, entry_t **data);

list_t *other_add_back(list_t *l, void *data);

*/
/* concatenation */
list_t *string_list_concat(const list_t *target, const list_t *src);

list_t *num_list_concat(const list_t *target, const list_t *src);

list_t *graph_list_concat(const list_t *target, const list_t *src);

list_t *node_list_concat(const list_t *target, const list_t *src);

/*
void dict_list_concat(list_t *target, const list_t *src);

*/
/* comparison */
int string_list_equals(const list_t *a, const list_t *b);

int num_list_equals(const list_t *a, const list_t *b);

int node_list_equals(const list_t *a, const list_t *b);

int graph_list_equals(const list_t *a, const list_t *b);

/*
int other_list_equals(const list_t *a, const list_t *b, int (*comp)(void *a,
    void *b));

*/

/* dequeue/pop */

list_t *pop(list_t *l);

/* peek */
void *peek(list_t *l);

/* freeing */
void free_list(list_t *r);

/*
void string_free_list(list_t *r);

void graph_free_list(list_t *r);

void node_free_list(list_t *r);
*/

/*
void dict_free_list(list_t *r);

void other_free_list(list_t *r);

*/
/* other */
list_t *range(int a, int b);

void *list_access(const list_t *l, int i);

void print_range(list_t *r);

void print_strings(list_t *r);

void free_list(list_t *r);

void free_range(list_t *r);

void num_index_insert(list_t *l, int i, float *a);

void string_index_insert(list_t *l, int i, char *a);

void node_index_insert(list_t *l, int i, node_t *a);

void graph_index_insert(list_t *l, int i, graph_t *a);

```

```c
float num_list_min(list_t *l);

float num_list_max(list_t *l);

char *string_list_min(list_t *l);

char *string_list_max(list_t *l);

int list_len(list_t *l);
```

Listing 28: list.h

B.6 list.c

```c
#include "dict.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

static list_t *add_front(list_t *l, void *data, void *(*copy)(void *src)) {
   list_t *new_node = (list_t *) malloc(sizeof(list_t));
   if(copy)
      new_node->data = copy(data);
   else
      new_node->data = data;
   new_node->previous = NULL;
   new_node->next = l;
   if(l)
      l->previous = new_node;
   return new_node;
}

static void *void_strcpy(void *src) {
   int len = strlen(src);
   char *target = malloc(sizeof(char) * len + 1);
   strncpy((char *) target, (char *) src, len + 1);
   target[len] = 0;
   return (void *) target;
}

list_t *string_add_front(list_t *l, char *data) {
   return add_front(l, (void *) data, void_strcpy);
}

static void *float_copy(void *src) {
   float *dst = malloc(sizeof(float));
   *dst = *(float *) src;
   return dst;
}

list_t *num_add_front(list_t *l, float *data) {
   return add_front(l, (void *) data, float_copy);
}

```

```c
static void *void_graph_copy(void *src) {
   return (void *) graph_copy(src);
}

list_t *graph_add_front(list_t *l, graph_t *data) {
   return add_front(l, (void *) data, void_graph_copy);
}

list_t *node_add_front(list_t *l, node_t *data) {
   return add_front(l, (void *) data, NULL);
}

/*
static void *void_dict_copy(entry_t **src) {
   return dict_copy(src);
}

list_t *dict_add_front(list_t *l, entry_t **data) {
   return add_front(l, (void *) data, void_dict_copy);
}

list_t *other_add_front(list_t *l, void *data) {
   return add_front(l, data, NULL);
}
*/

static list_t *add_back(list_t *l, void *data, void *(*copy)(void *src)) {
   list_t *new_node = (list_t *) malloc(sizeof(list_t));
   if(copy)
      new_node->data = copy(data);
   else
      new_node->data = data;
   new_node->next = NULL;
   if(l) {
      list_t *temp = l;
      while(temp->next)
         temp = temp->next;
      temp->next = new_node;
      new_node->previous = temp;
      return l;
   }
   new_node->previous = NULL;
   return new_node;
}

list_t *string_add_back(list_t *l, char *data) {
   return add_back(l, data, void_strcpy);
}

list_t *num_add_back(list_t *l, float *data) {
   return add_back(l, data, float_copy);
}

list_t *graph_add_back(list_t *l, graph_t *data) {
   return add_back(l, data, void_graph_copy);
}

list_t *node_add_back(list_t *l, node_t *data) {
   return add_back(l, data, NULL);
}

list_t *pop(list_t *l) {
   printf("l is null");
  if(!l) {
   return NULL;
  }
   list_t *head = l->next;
  if(head)
    head->previous = NULL;
   return head;
}

void *peek(list_t *l) {
   return l->data;
}

list_t *range(int a, int b) {
   list_t *r = NULL;
   int i;
   list_t *t;
   float *j;
   int sign = a > b ? 1 : -1;
   for(i = b; i != a + sign; i += sign) {
      t = (list_t *) malloc(sizeof(list_t));
      j = (float *) malloc(sizeof(float));
      *j = (float) i;
      t->data = j;
      if(r)
         r->previous = t;
      t->next = r;
      r = t;
   }
   return r;
}

void free_range(list_t *r) {
   if(r->next)
   for(r = r->next; r->next; r = r->next) {
      free(r->previous->data);
      free(r->previous);
   }
   free(r->data);
   free(r);
}

void print_range(list_t *r) {
   printf("[");
   for(; r; r = r->next)
```

```
      if(r->next)
         printf("%g, ", *((float *) r->data));
      else
         printf("%g", *((float *) r->data));
   printf("]\n");
}

void print_strings(list_t *r) {
   printf("[");
   for(; r; r = r->next)
      if(r->next)
         printf("%s, ", ((char *) r->data));
      else
         printf("%s]\n", ((char *) r->data));
}

void free_list(list_t *r) {
   if(r->next)
      for(r = r->next; r->next; r = r->next) {
         free(r->previous->data);
         free(r->previous);
      }
   else
      free(r->data);
   free(r);
}

list_t *num_list_copy(const list_t *src) {
   if(!src)
      return NULL;
   list_t *ret = NULL;
   const list_t *temp;
   for(temp = src; temp; temp = temp->next) {
      ret = num_add_back(ret, (float *) temp->data);
   }
   return ret;
}

list_t *string_list_copy(const list_t *src) {
   if(!src)
      return NULL;
   list_t *ret = NULL;
   const list_t *temp;
   for(temp = src; temp; temp = temp->next) {
      ret = string_add_back(ret, (char *) temp->data);
   }
   return ret;
}

list_t *node_list_copy(const list_t *src) {
   if(!src)
      return NULL;
   list_t *ret = NULL;
   const list_t *temp;
   for(temp = src; temp; temp = temp->next) {
      ret = node_add_back(ret, (node_t *) temp->data);
   }
   return ret;
}

list_t *graph_list_copy(const list_t *src) {
   if(!src)
      return NULL;
   list_t *ret = NULL;
   const list_t *temp;
   for(temp = src; temp; temp = temp->next) {
      ret = graph_add_back(ret, (graph_t *) temp->data);
   }
   return ret;
}

list_t *num_list_concat(const list_t *target, const list_t *src) {
   list_t *new_list = num_list_copy(target);
   if(new_list) {
      list_t *temp;
      for(temp = new_list; temp->next; temp = temp->next);
      temp->next = num_list_copy(src);
      return new_list;
   }
   return num_list_copy(src);
}

list_t *string_list_concat(const list_t *target, const list_t *src) {
   list_t *new_list = num_list_copy(target);
   if(new_list) {
      list_t *temp;
      for(temp = new_list; temp->next; temp = temp->next);
      temp->next = num_list_copy(src);
      return new_list;
   }
   return num_list_copy(src);
}

list_t *node_list_concat(const list_t *target, const list_t *src) {
   list_t *new_list = num_list_copy(target);
   if(new_list) {
      list_t *temp;
      for(temp = new_list; temp->next; temp = temp->next);
      temp->next = num_list_copy(src);
      return new_list;
   }
   return num_list_copy(src);
}

list_t *graph_list_concat(const list_t *target, const list_t *src) {
   list_t *new_list = num_list_copy(target);
   if(new_list) {
      list_t *temp;
      for(temp = new_list; temp->next; temp = temp->next);
      temp->next = num_list_copy(src);
      return new_list;
   }
   return num_list_copy(src);
}

void *list_access(const list_t *l, int i) {
   const list_t *temp;
   int j = 0;
   for(temp = l; temp; temp = temp->next) {
      if(j == i)
         return temp->data;
      j++;
   }
   return NULL;
}

int string_list_equals(const list_t *a, const list_t *b) {
   const list_t *temp_a = a;
   const list_t *temp_b = b;
   while(temp_a && temp_b) {
      if(strcmp((char *) temp_a->data, (char *) temp_b->data))
         return 0;
      temp_a = temp_a->next;
      temp_b = temp_b->next;
   }
   if(temp_a || temp_b)
      return 0;
   return 1;
}

int num_list_equals(const list_t *a, const list_t *b) {
   const list_t *temp_a = a;
   const list_t *temp_b = b;
   while(temp_a && temp_b) {
      if(!float_equals(*(float *) temp_a->data, *(float *) temp_b->data))
         return 0;
      temp_a = temp_a->next;
      temp_b = temp_b->next;
   }
   if(temp_a || temp_b)
      return 0;
   return 1;
}

int node_list_equals(const list_t *a, const list_t *b) {
   const list_t *temp_a = a;
   const list_t *temp_b = b;
   while(temp_a && temp_b) {
      if(temp_a->data != temp_b->data)
         return 0;
      temp_a = temp_a->next;
      temp_b = temp_b->next;
```

```c
    }
    if(temp_a || temp_b)
        return 0;
    return 1;
}

int graph_list_equals(const list_t *a, const list_t *b) {
    const list_t *temp_a = a;
    const list_t *temp_b = b;
    while(temp_a && temp_b) {
        if(!graph_equals((graph_t *) temp_a->data, (graph_t *) temp_b->data))
            return 0;
        temp_a = temp_a->next;
        temp_b = temp_b->next;
    }
    if(temp_a || temp_b)
        return 0;
    return 1;
}

int other_list_equals(const list_t *a, const list_t *b, int (*comp)(void *a,
    void *b)) {
    const list_t *temp_a = a;
    const list_t *temp_b = b;
    while(temp_a && temp_b) {
        if(comp(temp_a->data, temp_b->data))
            return 0;
        temp_a = temp_a->next;
        temp_b = temp_b->next;
    }
    if(temp_a || temp_b)
        return 0;
    return 1;
}

static void index_insert(list_t *l, int i, void *data, void *(*copy)(void *src
    )) {
    int j = 0;
    while(j < i) {
        j++;
        if(l->next)
            l = l->next;
        else
            break;
    }
    if(l->next) {
        if(copy)
            l->data = copy(data);
        else
            l->data = data;
    } else if(j == i) {
        l->next = (list_t *) malloc(sizeof(list_t));
        l->next->next = NULL;
        l->next->previous = l;
```

```
      if(copy)
         l->next->data = copy(data);
      else
         l->next->data = data;
   }
}

void num_index_insert(list_t *l, int i, float *a) {
   index_insert(l, i, a, float_copy);
}

void string_index_insert(list_t *l, int i, char *a) {
   index_insert(l, i, a, void_strcpy);
}

void node_index_insert(list_t *l, int i, node_t *a) {
   index_insert(l, i, a, NULL);
}

void graph_index_insert(list_t *l, int i, graph_t *a) {
   index_insert(l, i, a, void_graph_copy);
}

float num_list_min(list_t *l) {
   float min = (float) *(float *) l->data;
   for(; l; l = l->next)
      if(*(float *) l->data < min)
         min = (float) *(float *) l->data;
   return min;
}

float num_list_max(list_t *l) {
   float max = (float) *(float *) l->data;
   for(; l; l = l->next)
      if(*(float *) l->data > max)
         max = (float) *(float *) l->data;
   return max;
}

char *string_list_min(list_t *l) {
   char *min = (char *) l->data;
   for(; l; l = l->next)
      if(strcmp((char *)l->data, min) < 0)
         min = (char *) l->data;
   return min;
}

char *string_list_max(list_t *l) {
   char *max = (char *) l->data;
   for(; l; l = l->next)
      if(strcmp((char *)l->data, max) > 0)
         max = (char *) l->data;
   return max;
}
```

```c

int list_len(list_t *l) {
   int len = 0;
   for(; l; l = l->next)
      len++;
   return len;
}
```
Listing 29: list.c

B.7 main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "dict.h"

void print_nodes(graph_t *g) {
   list_t *n;
   for(n = g->nodes; n; n=n->next)
      printf("%s\n", (char *)((node_t *) n->data)->data);
}

int main() {
   char *a = malloc(12*sizeof(char));
   char *b = malloc(12*sizeof(char));
   char *c = malloc(12*sizeof(char));
   strncpy(a, "n: lo world", 12);
   strncpy(b, "m: hello bb", 12);
   strncpy(c, "o: fsf sefs", 12);

   node_t *n = init_node((void *) a);
   node_t *m = init_node((void *) b);
   node_t *o = init_node((void *) c);
   graph_t *g = NULL; // = init_graph();
   graph_t *h = init_graph();
   printf("%o\n", (int) a);
   printf("%o\n", (int) b);
   printf("%o\n", (int) c);

   g = graph_plus_node(g, n);
   add_node(g, m);
   add_node(h, o);

   graph_t *sum = node_plus_node(n, m);

   printf("print_nodes(g):\n");
   print_nodes(g);
   printf("print_nodes(h):\n");
   print_nodes(h);
   printf("print_nodes(sum):\n");
   print_nodes(sum);

   connect_dir_weighted(m, n, 0.4);
```

```c
  connect_dir_weighted(n, m, 0.8);
  printf("m --[%g]-- n\n", *(float *) get_node(m->out, n));
  printf("n --[%g]-- m\n", *(float *) get_node(n->out, m));

  /*
    printf("%s -> %s\n", (char *)m->data, (char *)m->out->node->data);
    printf("%s -> %s\n", (char *)n->data, (char *)n->out->node->data);
   */

  entry_t **d = init_dict();
  put_graph(d, g, (void *) "this is graph g's data");
  printf("here:\n");
  put_graph(d, h, "this is graph h");

  printf("%s\n", get_graph(d, g));
  printf("%s\n", get_graph(d, h));

  remove_undir_edge(m, n);
  /*
    printf("%s -> %x\n", (char *)m->data, (int)m->out);
    printf("%s -> %x\n", (char *)n->data, (int)n->out);
   */

  plus_equals(g, h);
  printf("print_nodes(g):\n");
  print_nodes(g);
  printf("print_nodes(h):\n");
  print_nodes(h);
  graph_t *g_copy = graph_copy(g);
  int x = graph_equals(g, g_copy);
  printf("x = %d\n", x);
  x = graph_equals(h, g_copy);
  printf("x = %d\n", x);
  printf("print_nodes(copy):\n");
  print_nodes(g_copy);
  graph_t *diff;

  diff = minus(g, h);
  remove_node(g, o);
  printf("print_nodes(g):\n");
  print_nodes(g);

  printf("done printing\n");

  // free everything
  free_graph(g);
  printf("g freed\n");
  free_graph(h);
  printf("h freed\n");
  free_node(m);
  printf("m freed\n");
  free_node(n);
  printf("n freed\n");
  free_node(o);
```

```c
    printf("o freed\n");

    return 0;
}
```

Listing 30: main.c

B.8 test.c

```c
#include <stdio.h>
#include "dict.h"

int main() {
    list_t *l = range(0, 10);
    print_range(l);
    free_range(l);
    l = range(10, 0);
    print_range(l);
    //free_range(l);
    float a = 30;
    list_t *m = range(20, 30);
    list_t *concat = num_list_concat(l, m);
    printf("concat: ");
    print_range(concat);
    printf("concat.len = %d\n", list_len(concat));
    float minf = num_list_min(concat);
    float maxf = num_list_max(concat);
    printf("min: %f\nmax: %f\n", minf, maxf);
    free_range(l);
    free_range(m);
    l = range(20, 20);
    print_range(l);
    free_range(l);

    char *s1 = "hello ";
    char *s2 = "world";
    l = NULL;
    l = string_add_front(l, s2);
    l = string_add_front(l, s1);
    char *min = string_list_min(l);
    char *max = string_list_max(l);
    printf("min: %s\nmax: %s\n", min, max);
    print_strings(l);
    free_list(l);
    printf("list done\n");

    l = NULL;
    l = string_add_back(l, s1);
    l = string_add_back(l, s2);
    list_t *string_copy = string_list_copy(l);
    printf("l = ");
    print_strings(l);
    printf("string_copy = ");
    print_strings(string_copy);

    printf("l[0] = %s, l[1] = %s\n", list_access(l, 0), list_access(l, 1));

    free_list(l);

    entry_t **d = NULL; //init_dict();
    d = put_string(d, "hello", (void *) "world");
    put_string(d, "hello", (void *) "world2");
    put_string(d, "elloh", (void *) "orldw");
    put_string(d, "something else", (void *) "new value");
    min = string_dict_min(d);
    max = string_dict_max(d);
    printf("min: %s: %s\nmax: %s: %s\n", min, (char *) get_string(d, min), max,
         get_string(d, max));
    char *got = (char *) get_string(d, "hello");
    printf("got %s\n", got);
    got = (char *) get_string(d, "elloh");
    printf("got %s\n", got);
    got = (char *) get_string(d, "something else");
    printf("got %s\n", got);
    got = (char *) get_string(d, "world");
    printf("got %s\n", got);
    printf("d.len = %d\n", dict_len(d));

    printf("1.23 == 1.24: %d\n", float_equals(1.23,1.24));

    entry_t **nums = init_dict();
    put_num(nums, 1.23, (void *) "1.23 val");
    put_num(nums, 1.23, (void *) "1.23 second val");
    put_num(nums, 1.24, (void *) "1.24 val");
    printf("nums.len = %d\n", dict_len(nums));
    minf = *(float *) num_dict_min(nums);
    maxf = *(float *) num_dict_max(nums);
    printf("min: %f\nmax: %f\n", minf, maxf);
    got = (char *)get_num(nums, 1.23);
    printf("got %s\n", got);
    got = (char *)get_num(nums, 1.24);
    printf("got %s\n", got);
    printf("removed 1.23:\n");
    num_dict_remove(nums, 1.23);
    got = (char *)get_num(nums, 1.23);
    printf("got %s\n", got);

    node_t *n = init_node("this is a node!");
    node_t *n2 = init_node("this is a node's value!");

    entry_t **other = init_dict();
    put_other(other, n, n2);
    node_t *g = get_other(other, n);
    int size = dict_len(other);
    printf("got node containing %s, size = %d\n", g->data, size);
    node_dict_remove(other, n);
    size = dict_len(other);
    g = get_other(other, n);
```

```
  printf("got node containing %x\n, size = %d\n", (unsigned long) g, size);

 entry_t **empty = init_dict();
 int len = dict_len(empty);


  return 0;
}
```

Listing 31: test.c

B.9 snippets.c

This file was just a few notes on what C code was for a semantically equivalent line of d.o.t.s. code.

```
/* for node in graph */
list_t *temp;
node_t *node;
for(temp = graph->nodes; temp; temp = temp->next) {
   node = temp->data;
   /* loop body */
}

/* for node in list */
list_t *node;
for(node = list; node; node = node->next) {
   /* loop body */
}

/* string s = arg */
char *s = arg;

/* num n = arg */
float n = arg;

/* node x */
node_t *x = init_node("");

/* node y(str) */
node_t *x = init_node(string);

/* graph g1; */
graph_t *g1 = init_graph();

/* graph g2 = g1; */
graph_t *g2 = graph_copy(g1);

/* graph g3 = {
     x
     y
  };
  */
graph_t *g3 = init_graph();
add_node(g3, x);
add_node(g3, y);

/* function declarations:
  def return_type function_name(num arg1, node arg2) {
     return return_type;
  }
  */

return_type function_name(float arg1, node_t *arg2) {
   return return_type;
}

/* node1 == node 2 */
node_compare(node1, node2);

/* node1 != node2 */
!node_compare(node1, node2);

/* while statement */
while(statement) {
   /* loop body */
}

/* if condition */
if(condition) {

   /* else if */
} else if {

   /* else */
} else {
}

/* x -- y */
connect_undir(x, y);

/* x --> y */
connect_dir(x, y);

/* x --[n] y */
connect_undir(x, y, n);

/* x -->[n] y */
connect_dir(x, y, n);

/* x [m]--[n] y */
connect_dir(x, y, n);
connect_dir(y, x, m);

/* g1 = g2 + g3 */
g1 = plus(g2, g3);

/* g1 += g2 */
plus_equals(g1, g2);
```

```

/* list<num> l = [1, 2, 3] */
list_t *l = NULL;
int *i;

i = (int *) malloc(sizeof(int));
*i = 1;
l = add_back(l, i);
i = (int *) malloc(sizeof(int));
*i = 2;
l = add_back(l, i);
i = (int *) malloc(sizeof(int));
*i = 3;
l = add_back(l, i);

/** dict initialization **/
/* dict<type, type> d; */
entry_t **d = init_dict();

/** dict insertion **/
/* d["literal"] = something */
put_string(d, "literal", (void *) &something);

/* d[1.23] = something */
put_num(d, 1.23, (void *) &something);

/* d[_node] = something */
put_node(d, (void *) &_node, (void *) &something);

/** dict access **/
/* something = d["key"]; */
something = *(type *) get_string(d, "key");

/* something = d[1.23]; */
something = *(type *) get_num(d, 1.23);

/* something = d[_node]; */
something = *(type *) get_other(d, &_node);

/* for key in d */
// d[i] --> (*d)[i] OR d[0][i]
int i;
entry_t *temp;
void *key;
for(i = 0; i < TABLE_SIZE; i++) {
   for(temp = d[i]; temp; temp = temp->next) {
      key = temp->key;

   }
}

/** printing dicts **/
/** print d **/
int i;
```

```
entry_t *temp;
void *key;
/* print "{"; */
int first = 1;
for(i = 0; i < TABLE_SIZE; i++) {
   for(temp = d[i]; temp; temp = temp->next) {
      key = temp->key;
      if(first) {
         first = 0;
         /* print key, ": ", value */
      } else {
         /* print ", " , key, ": ", value */
      }
   }
}
/* print "}\n" */

/** printing lists **/
/** print list_1; **/
list_t *temp;
int first = 1;
/* print "[" */
for(temp = list_1; temp; temp = temp->next) {
   if(first) {
      first = 0;
      /* print temp->data */
   } else {
      /* print ", ", temp->data */
   }
}
/* print "]\n" */

/** adding strings **/
/* s3 = s1 + s2; */
int len = strlen(s1) + strlen(s2) + 1;
char *s3 = (char *) calloc(len, sizeof(char));
strncpy(s3, s1, strlen(s1));
strncpy(s3, s2, strlen(s2));

/** removals */
/** dict - key1 **/
int i;
entry_t *temp;
void *key;
for(i = 0; i < TABLE_SIZE; i++) {
   for(temp = d[i]; temp; temp = temp->next) {
      key = temp->key;
      if(/*key == key1*/) {
         dict_remove(d[i], temp);
         i = TABLE_SIZE;
         temp = NULL;
      }
   }
}
```

```

/* list.enqueue(data) */
<type>_add_back(list, data);

/* list.push(data) */
<type>_add_front(list, data);

/* list l2 = l1 */
list_t *l2 = <type>_list_copy(l1);

/* list l3 = l1 + l2 */
list_t *l3 = <type>_list_concat(l1, l2);

/* data = list.peek() */
<type> *data = (<type> *) peek(list);

/** ONLY removes first element from list and discards data **/
/* list.pop() */
list = pop(list);

/* something = list[i] */
something = list_access(list, i);

/* g1 = g2 - g3 */
graph_t *g1 = minus(g2, g3);

/* list list_1 = list_2 + list_3 */
list_t *list_1 = <type>_list_concat(list_2, list_3);

/** graph = node + node **/
/* graph g2 = n1 + n2 */
graph_t *g2 = node_plus_node(n1, n2);

/** graph = graph + node **/
/* graph g2 = g1 + n2 */
graph_t *g2 = graph_plus_node(g1, n2);

/** dict.remove(key) **/
<type>_dict_remove(dict, key);

/* dict.min() */
<type>_dict_min(dict);

/* list.max() */
<type>_list_max(list);

/** list[i] = something **/
<type>_index_insert(list, i, something);
```
Listing 32: snippets.c

C Makefiles

C.1 src/Makefile

```
OBJS = ast.cmo Sast.cmo parser.cmo scanner.cmo translate.cmo analyzer.cmo typeConverter.cmo compile.cmo

TESTS = \
arith1 \
arith2 \
fib \
for1 \
func1 \
func2 \
func3 \
gcd \
gcd2 \
global1 \
hello \
if1 \
if2 \
if3 \
if4 \
ops1 \
var1 \
while1

# Choose one
YACC = ocamlyacc -v
# YACC = menhir --explain

TARFILES = Makefile testall.sh scanner.mll parser.mly \
	ast.ml Sast.ml bytecode.ml interpret.ml compile.ml execute.ml microc.ml \
	$(TESTS:%=tests/test-%.mc) \
	$(TESTS:%=tests/test-%.out)


build: analyzer clib

.PHONY: setup
setup:
	cd clib ; make clean
	cd clib ; make library

clib:
	make -f clib/Makefile

analyzer : str $(OBJS)
	ocamlc -o dotc str.cma $(OBJS)

ast_print : str $(OBJS)
	ocamlc -o dotc str.cma $(OBJS) astPrinter.ml

```

```
str :
	ocamlopt str.cmxa

microc : $(OBJS)
	ocamlc -o microc $(OBJS)

.PHONY : test
test : microc testall.sh
	./testall.sh

scanner.ml : scanner.mll
	ocamllex scanner.mll

parser.ml parser.mli : parser.mly
	$(YACC) parser.mly

%.cmo : %.ml
	ocamlc -c $<

%.cmi : %.mli
	ocamlc -c $<

microc.tar.gz : $(TARFILES)
	cd .. && tar czf microc/microc.tar.gz $(TARFILES:%=microc/%)

.PHONY : clean
clean :
	rm -f dotc microc parser.output parser.automaton parser.ml parser.mli
	    scanner.ml testall.log \
	*.cmo *.cmi *.out *.diff exec compile.c

# Generated by ocamldep *.ml *.mli

analyzer.cmo: sast.cmo ast.cmo
analyzer.cmx: sast.cmx ast.cmx
generator.cmo: sast.cmo
generator.cmx: sast.cmx
parser.cmo: ast.cmo parser.cmi
parser.cmx: ast.cm parser.cmi
dot.cmo: scanner.cmo sast.cmo parser.cmi ast.cmo analyzer.cmo
dot.cmx: scanner.cmx sast.cmo parser.cmx ast.cmx analyzer.cmx
sast.cmo: ast.cmo
sast.cmx: ast.cmx
scanner.cmo: parser.cmi
scanner.cmx: parser.cmx
parser.cmi: ast.cmo
str.cma: str.cmxa
```

Listing 33: src/Makefile

C.2 src/clib/Makefile

```
CC = gcc

```

```
CFLAGS = -Wall

CFILES= graph.c node.c list.c dict.c

main: main.o graph.o node.o dict.o

main.o: main.c graph.h node.h

graph.o: graph.h node.h graph.c

node.o: node.h node.c dict.o

test: test.o list.o dict.o graph.o node.o

list.o: list.c list.h

test.o: list.h test.c

dict.o: dict.c dict.h

.PHONY: objects
objects:
	$(CC) -c $(CFILES)

.PHONY: library
library: objects
	ar -cvq libdots.a *.o

.PHONY: clean
clean:
	rm -f *.o main test *.a
```
Listing 34: src/clib/Makefile

D Test Suite

D.1 runtest.py

This was the main test suite script. It tested the whole compilation process.

```python
# Test automation script

import os, sys, glob
import argparse
from subprocess32 import check_output, Popen, PIPE, call

#####################
# ARGUMENT PARSING: #
#####################

parser = argparse.ArgumentParser(description='Run tests on compilation of dots
    files.')
parser.add_argument("-c", "--clean", action="store_true",
    help="removes all created files after tests have finished")
args = parser.parse_args()

#########
# TESTS #
#########

path = r'dtest'
npath = r'ntests'

print "Using positive testing dir: " + path
print "Using negative testing dir: " + npath

summary_results = {}
summary_results_n = {}

for directory in os.walk(path):
   # walk through the test directory
   print ('\nRunning tests in "' + directory[0] + '" folder:')
   print ('**************************************************')
   for dir_entry in os.listdir(directory[0]):
      filepath = os.path.join(directory[0], dir_entry)
      if os.path.isfile(filepath) and filepath[-5:] == '.dots':
         print('\nRunning tests: ' + dir_entry)
         print('================================')

         comp_success = False
         try:
            return_code = call(['./gdc', filepath, os.path.join(directory[0],
                dir_entry[:-5] + '.exec')],
               timeout=15)
            if return_code == 0:
               print 'COMPILATION SUCCESSFUL'
               comp_success = True
            else:
               print 'COMPILATION FAILED'
               summary_results[dir_entry[:-5]] = ('fail', directory[0])
         except:
            print 'compile executable. Stop.'
            continue;

         if (comp_success):
            out_child = Popen('./' + os.path.join(directory[0], dir_entry
                [:-5]) + '.exec',
               shell=True, stdout=PIPE, stderr=PIPE)
            output = out_child.communicate()[0]

            output_filepath = os.path.join(directory[0], dir_entry[:-5] + '.
                outgdc')
            with open(output_filepath, 'w') as intermediate_output:
               intermediate_output.write(output)

            out_filepath = os.path.join(directory[0], dir_entry[:-5] + '.out')
            output_filepath = os.path.join(directory[0], dir_entry[:-5] + '.
               outgdc')

            if (os.path.exists(out_filepath)):
               diff_command = ['diff', '-bB', out_filepath, output_filepath]
               diff_child = Popen(diff_command, stdout=PIPE)
               diff_output = diff_child.communicate()[0]

               if diff_output.strip() == '':
                  print 'PASSED TEST'
                  summary_results[dir_entry[:-5]] = ('pass', directory[0])
               else:
                  print 'FAILED TEST....writing diff files'
                  summary_results[dir_entry[:-5]] = ('fail', directory[0])
                  with open(os.path.join(directory[0], dir_entry[:-5] + '.dif'
                     ), 'w') as output_diff:
                     output_diff.write(diff_output.strip())
            else:
               print "FAIL: no .out file exists to check against"


for directory in os.walk(npath):
   print ('\nRunning tests in "' + directory[0] + '" folder:')
   print ('*************************************************')
   for dir_entry in os.listdir(directory[0]):
      filepath = os.path.join(directory[0], dir_entry)
      if os.path.isfile(filepath) and filepath[-5:] == '.dots':
         print('\nRunning tests: ' + dir_entry)
         print('================================')

         comp_success = False
         try:
            return_code = call(['./gdc', filepath, os.path.join(directory[0],
               dir_entry[:-5] + '.exec')],
               timeout=30)
            if return_code == 0:
               print 'COMPILATION SUCCESSFUL'
               comp_success = True
            else:
               print 'COMPILATION FAILED'
               summary_results_n[dir_entry[:-5]] = ('fail', directory[0])
         except:

            continue;

         if (comp_success):
            out_child = Popen('./' + os.path.join(directory[0], dir_entry
               [:-5]) + '.exec',
               shell=True, stdout=PIPE)
            output = out_child.communicate()[0]

            output_filepath = os.path.join(directory[0], dir_entry[:-5] + '.
               outgdc')
            with open(output_filepath, 'w') as intermediate_output:
                intermediate_output.write(output)

            out_filepath = os.path.join(directory[0], dir_entry[:-5] + '.out')
            output_filepath = os.path.join(directory[0], dir_entry[:-5] + '.
                outgdc')

            if (os.path.exists(out_filepath)):
                diff_command = ['diff', '-bB', out_filepath, output_filepath]
                diff_child = Popen(diff_command, stdout=PIPE)
                diff_output = diff_child.communicate()[0]

                if diff_output.strip() == '':
                    print 'PASSED TEST'
                    summary_results_n[dir_entry[:-5]] = ('pass', directory[0])
                else:
                    print 'FAILED TEST....writing diff files'
                    summary_results_n[dir_entry[:-5]] = 'fail'
                    with open(os.path.join(directory[0], dir_entry[:-5] + '.dif'
                        ), 'w') as output_diff:
                        output_diff.write(diff_output.strip())
            else:
                print "FAIL: no .out file exists to check against"

print('\n Tests completed.')
print('\n Summary below (checked boxes = performed as expected): \n')

#################
# PRINT SUMMARY #
#################
print('Tests that should pass:')
for test_name in sorted(summary_results):
    if summary_results[test_name][0] == 'pass':
        print('[X] ' + test_name + '(' + summary_results[test_name][1] + ')')
    else:
        print('[ ] ' + test_name + '(' + summary_results[test_name][1] + ')')


print('\nTests that should fail:')
for test_name in sorted(summary_results_n):
    if summary_results_n[test_name][0] == 'pass':
        print('[ ] ' + test_name + '(' + summary_results_n[test_name][1] + ')')
    else:
        print('[X] ' + test_name + '(' + summary_results_n[test_name][1] + ')')

############
# CLEAN-UP #
############

# remove all the intermediate file output if the clean flag is set
if args.clean:
    file_exts = ['*.outgdc', '*.dif', '*.c', '*.exec']
    for ext in file_exts:
        for directory in os.walk(path):
            for f in glob.glob(os.path.join(directory[0], ext)):
```

```
            print(directory[0])
            os.remove(f)
      for directory in os.walk(npath):
         for f in glob.glob(os.path.join(directory[0], ext)):
            os.remove(f)
```

Listing 35: runtest.py

D.2 testall.py

This was an alternate test script for testing using menhir.

```
# Menhir test automating script
# Might use it to automate more tests
# Make sure all of the input files are .txt files!

import os, sys, subprocess
run_normal = True
suppress_stderr = True

if len(sys.argv) == 1:
   print('Running with default configurations:\n')
elif '-f' in sys.argv and '-e' in sys.argv:
   run_normal = False
   suppress_stderr = False
elif '-f' in sys.argv:
   run_normal = False
elif '-e' in sys.argv:
   suppress_stderr = False
else:
   print('usage: -f, or no command line args')
   print('-f: prints full results of every test')
   print('-e: show stderr of menhir')
   print ('no command line args: runs all tests and only prints tests that
      failed, suppresses stderr of menhir.\n')
   sys.exit()

path = r'menhir-tests'
for dir_entry in os.listdir(path):
   filepath = os.path.join(path, dir_entry)
   if os.path.isfile(filepath) and filepath[-3:] == 'txt':
      print('Running tests in ' + dir_entry)
      print('================================\n')
      with open(filepath, 'r') as test_file:
         for line in test_file:
            supposed_to_pass = True
            if line[:3] == '***' or line[0] == '\n':
               continue

            to_pipe = line.strip().split()
            if to_pipe[0] == 'f**':
               supposed_to_pass = False
               to_pipe = to_pipe[1:]

            to_pipe.insert(0, 'echo')

            #can pipe with subprocess only by opening another process, not standard syntax
            menhir_input = subprocess.Popen(to_pipe, stdout=subprocess.PIPE)
            menhir_cmd = ['menhir', '--interpret', '--interpret-show-cst', 'parser.mly']
            if suppress_stderr == True:
               with open(os.devnull, 'w') as devnull:
                  output = subprocess.check_output(menhir_cmd, stdin=menhir_input.stdout, stderr=devnull)
            else:
               output = subprocess.check_output(menhir_cmd, stdin=menhir_input.stdout)

            if run_normal == False:
               print(output)
            else:
               if supposed_to_pass == True:
                  if 'REJECT' in output:
                     print('"' + line.strip() + '" failed when it should pass.\n')
               else:
                  if 'ACCEPT' in output:
                     print('"' + line.strip() + '" passed when it should fail.\n')

   else:
      print (dir_entry + ' is messed up.\n')
      print ('--------------------------\n\n')

print('\n Tests completed.')
```
Listing 36: testall.py

E Example Code Compiled C Programs

E.1 hello-world.dots.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dict.h>
node_t * v1;
char * v2;
int main (int argc, char ** argv)
{
char ** v3;
v3 = malloc(sizeof(char *));
*(v3) = malloc(strlen("miami") + 1);
strcpy(*(v3), "miami");
v1 = init_node("");
(v1)->data = *(v3);
char ** v4;
v4 = malloc(sizeof(char *));
*(v4) = malloc(strlen("Hello World!") + 1);
strcpy(*(v4), "Hello World!");
char ** v5;
v5 = &(v2);
*(v5) = *(v4);
char ** v6;
v6 = &(v2);
printf("%s", *(v6));
char ** v7;
v7 = malloc(sizeof(char *));
*(v7) = malloc(strlen("\n") + 1);
strcpy(*(v7), "\n");
printf("%s", *(v7));
node_t ** v8;
v8 = &(v1);
printf("%s", "N-");
printf("%d", (int)(*(v8)));
printf("%s", "(\"");
printf("%s", (char *)((*(v8))->data));
printf("\")");
char ** v9;
v9 = malloc(sizeof(char *));
*(v9) = malloc(strlen("\n") + 1);
strcpy(*(v9), "\n");
printf("%s", *(v9));;
}
```

Listing 37: hello-world.dots.c

E.2 sample01.dots.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dict.h>
node_t * v1;
node_t * v2;
node_t * v3;
node_t * v4;
list_t * v5 = NULL;
char * v6;
graph_t * v7 = NULL;
graph_t * v8 = NULL;
list_t * v9 = NULL;
char * f6 (list_t * v89, node_t * v90)
{
list_t ** v91;
v91 = &(v89);
node_t * v93;
list_t * v92 = NULL;
```

```
for (v92 = *(v91); v92; v92 = (v92)->next) {
v93 = *((node_t **)((v92)->data));
node_t ** v94;
v94 = &(v93);
node_t ** v95;
v95 = &(v90);
if (*(v94) == *(v95)) {
char ** v96;
v96 = malloc(sizeof(char *));
*(v96) = malloc(strlen("yes") + 1);
strcpy(*(v96), "yes");
return *(v96);
} else {

}
}
char ** v97;
v97 = malloc(sizeof(char *));
*(v97) = malloc(strlen("no") + 1);
strcpy(*(v97), "no");
return *(v97);
}

char * f7 (graph_t * v89, node_t * v90)
{
graph_t ** v91;
v91 = &(v89);
node_t * v93;
list_t * v92 = NULL;
for (v92 = (*v91)->nodes; v92; v92 = (v92)->next) {
v93 = (node_t *)((v92)->data);
node_t ** v94;
v94 = &(v93);
node_t ** v95;
v95 = &(v90);
if (*(v94) == *(v95)) {
char ** v96;
v96 = malloc(sizeof(char *));
*(v96) = malloc(strlen("yes") + 1);
strcpy(*(v96), "yes");
return *(v96);
} else {

}
}
char ** v97;
v97 = malloc(sizeof(char *));
*(v97) = malloc(strlen("no") + 1);
strcpy(*(v97), "no");
return *(v97);
}

list_t * f8 (graph_t * v89, graph_t * v90)
{
```

```
list_t * v91 = NULL;
graph_t ** v92;
v92 = &(v89);
node_t * v94;
list_t * v93 = NULL;
for (v93 = (*v92)->nodes; v93; v93 = (v93)->next) {
v94 = (node_t *)((v93)->data);
graph_t ** v95;
v95 = &(v90);
node_t * v97;
list_t * v96 = NULL;
for (v96 = (*v95)->nodes; v96; v96 = (v96)->next) {
v97 = (node_t *)((v96)->data);
node_t ** v98;
v98 = &(v94);
node_t ** v99;
v99 = &(v97);
if (*(v98) == *(v99)) {
list_t ** v100;
v100 = &(v91);
node_t ** v101;
v101 = &(v94);
list_t ** v102;
v102 = malloc(sizeof(void));
*(v100) = node_add_back(*(v100), *(v101));;
} else {

}
}
}
list_t ** v103;
v103 = &(v91);
return *(v103);
}

int main (int argc, char ** argv)
{
char ** v10;
v10 = malloc(sizeof(char *));
*(v10) = malloc(strlen("chicago") + 1);
strcpy(*(v10), "chicago");
v1 = init_node("");
(v1)->data = *(v10);
char ** v11;
v11 = malloc(sizeof(char *));
*(v11) = malloc(strlen("bar") + 1);
strcpy(*(v11), "bar");
v2 = init_node("");
(v2)->data = *(v11);
char ** v12;
v12 = malloc(sizeof(char *));
*(v12) = malloc(strlen("foo") + 1);
strcpy(*(v12), "foo");
v3 = init_node("");
```

```
(v3)->data = *(v12);
char ** v13;
v13 = malloc(sizeof(char *));
*(v13) = malloc(strlen("blah") + 1);
strcpy(*(v13), "blah");
v4 = init_node("");
(v4)->data = *(v13);
list_t ** v14;
v14 = malloc(sizeof(list_t *));
*(v14) = NULL;
list_t ** v18;
node_t ** v15;
v15 = &(v1);
*(v14) = node_add_back(*(v14), v15);
node_t ** v16;
v16 = &(v2);
*(v14) = node_add_back(*(v14), v16);
node_t ** v17;
v17 = &(v3);
*(v14) = node_add_back(*(v14), v17);
v18 = v14;
list_t ** v19;
v19 = &(v5);
*(v19) = *(v18);
char ** v20;
v20 = malloc(sizeof(char *));
*(v20) = malloc(strlen("list contains: \n") + 1);
strcpy(*(v20), "list contains: \n");
printf("%s", *(v20));
printf("[");
list_t ** v22;
v22 = &(v5);
node_t * v24;
list_t * v23 = NULL;
for (v23 = *(v22); v23; v23 = (v23)->next) {
v24 = *((node_t **)((v23)->data));
node_t ** v25;
v25 = &(v24);
printf("%s", "N-");
printf("%d", (int)(*(v25)));
printf("%s", "(\"");
printf("%s", (char *)((*(v25))->data));
printf("\")");;
char ** v26;
v26 = malloc(sizeof(char *));
*(v26) = malloc(strlen(", ") + 1);
strcpy(*(v26), ", ");
printf("%s", *(v26));;
}
printf("]");
char ** v27;
v27 = malloc(sizeof(char *));
*(v27) = malloc(strlen("\n") + 1);
strcpy(*(v27), "\n");
```

```
printf("%s", *(v27));
char ** v28;
v28 = malloc(sizeof(char *));
*(v28) = malloc(strlen("\n") + 1);
strcpy(*(v28), "\n");
printf("%s", *(v28));;
list_t ** v29;
v29 = &(v5);
node_t ** v30;
v30 = &(v1);
char ** v31;
v31 = malloc(sizeof(char *));
*(v31) = f6(*(v29), *(v30));
char ** v32;
v32 = &(v6);
*(v32) = *(v31);
node_t ** v33;
v33 = &(v1);
printf("%s", "N-");
printf("%d", (int)(*(v33)));
printf("%s", "(\"");
printf("%s", (char *)((*(v33))->data));
printf("\")");
char ** v34;
v34 = malloc(sizeof(char *));
*(v34) = malloc(strlen(" in node_list?\n") + 1);
strcpy(*(v34), " in node_list?\n");
printf("%s", *(v34));
char ** v35;
v35 = malloc(sizeof(char *));
*(v35) = malloc(strlen("\t") + 1);
strcpy(*(v35), "\t");
printf("%s", *(v35));
char ** v36;
v36 = &(v6);
printf("%s", *(v36));
char ** v37;
v37 = malloc(sizeof(char *));
*(v37) = malloc(strlen("\n") + 1);
strcpy(*(v37), "\n");
printf("%s", *(v37));;
list_t ** v38;
v38 = &(v5);
node_t ** v39;
v39 = &(v4);
char ** v40;
v40 = malloc(sizeof(char *));
*(v40) = f6(*(v38), *(v39));
char ** v41;
v41 = &(v6);
*(v41) = *(v40);
node_t ** v42;
v42 = &(v4);
printf("%s", "N-");
printf("%d", (int)(*(v42)));
printf("%s", "(\"");
printf("%s", (char *)((*(v42))->data));
printf("\")");
char ** v43;
v43 = malloc(sizeof(char *));
*(v43) = malloc(strlen(" in node_list?\n") + 1);
strcpy(*(v43), " in node_list?\n");
printf("%s", *(v43));
char ** v44;
v44 = malloc(sizeof(char *));
*(v44) = malloc(strlen("\t") + 1);
strcpy(*(v44), "\t");
printf("%s", *(v44));
char ** v45;
v45 = &(v6);
printf("%s", *(v45));
char ** v46;
v46 = malloc(sizeof(char *));
*(v46) = malloc(strlen("\n") + 1);
strcpy(*(v46), "\n");
printf("%s", *(v46));;
char ** v47;
v47 = malloc(sizeof(char *));
*(v47) = malloc(strlen("\n\n") + 1);
strcpy(*(v47), "\n\n");
printf("%s", *(v47));;
node_t ** v48;
v48 = &(v1);
node_t ** v49;
v49 = &(v4);
graph_t ** v50;
v50 = malloc(sizeof(graph_t *));
*(v50) = (node_plus_node(*(v48), *(v49)));
graph_t ** v51;
v51 = &(v7);
*(v51) = *(v50);
graph_t ** v52;
v52 = &(v7);
node_t ** v53;
v53 = &(v2);
graph_t ** v54;
v54 = malloc(sizeof(graph_t *));
*(v54) = (graph_plus_node(*(v52), *(v53)));
graph_t ** v55;
v55 = &(v7);
*(v55) = *(v54);
char ** v56;
v56 = malloc(sizeof(char *));
*(v56) = malloc(strlen("G1 contains:\n") + 1);
strcpy(*(v56), "G1 contains:\n");
printf("%s", *(v56));;
graph_t ** v57;
v57 = &(v7);
```

```
node_t * v59;
list_t * v58 = NULL;
for (v58 = (*v57)->nodes; v58; v58 = (v58)->next) {
v59 = (node_t *)((v58)->data);
node_t ** v60;
v60 = &(v59);
printf("%s", "N-");
printf("%d", (int)(*(v60)));
printf("%s", "(\"");
printf("%s", (char *)((*(v60))->data));
printf("\")");
char ** v61;
v61 = malloc(sizeof(char *));
*(v61) = malloc(strlen("\n") + 1);
strcpy(*(v61), "\n");
printf("%s", *(v61));;
}
char ** v62;
v62 = malloc(sizeof(char *));
*(v62) = malloc(strlen("\n") + 1);
strcpy(*(v62), "\n");
printf("%s", *(v62));;
v8 = init_graph();
float* v63;
v63 = malloc(sizeof(float));
*(v63) = 22.3;
connect_dir_weighted (v3, v1, *(v63));
connect_dir(v2, v3);
connect_undir(v1, v2);
add_node(v8, v1);
add_node(v8, v2);
add_node(v8, v2);
add_node(v8, v3);
add_node(v8, v3);
add_node(v8, v1);
char ** v64;
v64 = malloc(sizeof(char *));
*(v64) = malloc(strlen("G2 contains: \n") + 1);
strcpy(*(v64), "G2 contains: \n");
printf("%s", *(v64));;
graph_t ** v65;
v65 = &(v8);
node_t * v67;
list_t * v66 = NULL;
for (v66 = (*v65)->nodes; v66; v66 = (v66)->next) {
v67 = (node_t *)((v66)->data);
node_t ** v68;
v68 = &(v67);
printf("%s", "N-");
printf("%d", (int)(*(v68)));
printf("%s", "(\"");
printf("%s", (char *)((*(v68))->data));
printf("\")");
char ** v69;
v69 = malloc(sizeof(char *));
*(v69) = malloc(strlen("\n") + 1);
strcpy(*(v69), "\n");
printf("%s", *(v69));;
}
char ** v70;
v70 = malloc(sizeof(char *));
*(v70) = malloc(strlen("\n") + 1);
strcpy(*(v70), "\n");
printf("%s", *(v70));;
graph_t ** v71;
v71 = &(v7);
node_t ** v72;
v72 = &(v3);
char ** v73;
v73 = malloc(sizeof(char *));
*(v73) = f7(*(v71), *(v72));
char ** v74;
v74 = &(v6);
*(v74) = *(v73);
char ** v75;
v75 = malloc(sizeof(char *));
*(v75) = malloc(strlen("z in g1? ") + 1);
strcpy(*(v75), "z in g1? ");
printf("%s", *(v75));
char ** v76;
v76 = &(v6);
printf("%s", *(v76));
char ** v77;
v77 = malloc(sizeof(char *));
*(v77) = malloc(strlen("\n") + 1);
strcpy(*(v77), "\n");
printf("%s", *(v77));;
graph_t ** v78;
v78 = &(v8);
node_t ** v79;
v79 = &(v3);
char ** v80;
v80 = malloc(sizeof(char *));
*(v80) = f7(*(v78), *(v79));
char ** v81;
v81 = &(v6);
*(v81) = *(v80);
char ** v82;
v82 = malloc(sizeof(char *));
*(v82) = malloc(strlen("z in g2? ") + 1);
strcpy(*(v82), "z in g2? ");
printf("%s", *(v82));
char ** v83;
v83 = &(v6);
printf("%s", *(v83));
char ** v84;
v84 = malloc(sizeof(char *));
*(v84) = malloc(strlen("\n") + 1);
```

```
strcpy(*(v84), "\n");
printf("%s", *(v84));;
graph_t ** v85;
v85 = &(v7);
graph_t ** v86;
v86 = &(v8);
list_t ** v87;
v87 = malloc(sizeof(list_t *));
*(v87) = f8(*(v85), *(v86));
list_t ** v88;
v88 = &(v9);
*(v88) = *(v87);
char ** v89;
v89 = malloc(sizeof(char *));
*(v89) = malloc(strlen("\nSHARED NODES:\n") + 1);
strcpy(*(v89), "\nSHARED NODES:\n");
printf("%s", *(v89));;
}
```

Listing 38: sample01.dots.c

E.3 bst.dots.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dict.h>
graph_t * v1 = NULL;
node_t * v2;
node_t * v3;
node_t * v4;
node_t * v5;
node_t * v6;
node_t * v7;
list_t * v8 = NULL;
list_t * v9 = NULL;
entry_t** v10 = NULL;
node_t * v11;
float v12;
int f6 (list_t * v111, node_t * v112)
{
list_t ** v113;
v113 = &(v111);
node_t * v115;
list_t * v114 = NULL;
for (v114 = *(v113); v114; v114 = (v114)->next) {
v115 = *((node_t **)((v114)->data));
node_t ** v116;
v116 = &(v115);
node_t ** v117;
v117 = &(v112);
if (*(v116) == *(v117)) {
int* v118;
v118 = malloc(sizeof(int));
*(v118) = 1;
return *(v118);
} else {

}
}
int* v119;
v119 = malloc(sizeof(int));
*(v119) = 0;
return *(v119);
}

int main (int argc, char ** argv)
{
char ** v13;
v13 = malloc(sizeof(char *));
*(v13) = malloc(strlen("Searching\n") + 1);
strcpy(*(v13), "Searching\n");
printf("%s", *(v13));;
char ** v14;
v14 = malloc(sizeof(char *));
*(v14) = malloc(strlen("x") + 1);
strcpy(*(v14), "x");
v2 = init_node("");
(v2)->data = *(v14);
char ** v15;
v15 = malloc(sizeof(char *));
*(v15) = malloc(strlen("y") + 1);
strcpy(*(v15), "y");
v3 = init_node("");
(v3)->data = *(v15);
char ** v16;
v16 = malloc(sizeof(char *));
*(v16) = malloc(strlen("z") + 1);
strcpy(*(v16), "z");
v4 = init_node("");
(v4)->data = *(v16);
char ** v17;
v17 = malloc(sizeof(char *));
*(v17) = malloc(strlen("a") + 1);
strcpy(*(v17), "a");
v5 = init_node("");
(v5)->data = *(v17);
char ** v18;
v18 = malloc(sizeof(char *));
*(v18) = malloc(strlen("b") + 1);
strcpy(*(v18), "b");
v6 = init_node("");
(v6)->data = *(v18);
char ** v19;
v19 = malloc(sizeof(char *));
*(v19) = malloc(strlen("c") + 1);
strcpy(*(v19), "c");
v7 = init_node("");
```

```
(v7)->data = *(v19);
node_t ** v20;
v20 = &(v2);
node_t ** v21;
v21 = &(v3);
graph_t ** v22;
v22 = malloc(sizeof(graph_t *));
*(v22) = (node_plus_node(*(v20), *(v21)));
graph_t ** v23;
v23 = &(v1);
*(v23) = *(v22);
graph_t ** v24;
v24 = &(v1);
node_t ** v25;
v25 = &(v4);
graph_t ** v26;
v26 = malloc(sizeof(graph_t *));
*(v26) = (graph_plus_node(*(v24), *(v25)));
graph_t ** v27;
v27 = &(v1);
*(v27) = *(v26);
float* v28;
v28 = malloc(sizeof(float));
*(v28) = 2;
connect_dir_weighted (v2, v3, *(v28));
float* v29;
v29 = malloc(sizeof(float));
*(v29) = 1.5;
connect_dir_weighted (v2, v4, *(v29));
float* v30;
v30 = malloc(sizeof(float));
*(v30) = 4;
connect_dir_weighted (v4, v3, *(v30));
float* v31;
v31 = malloc(sizeof(float));
*(v31) = 2;
connect_dir_weighted (v3, v7, *(v31));
float* v32;
v32 = malloc(sizeof(float));
*(v32) = 2.5;
connect_dir_weighted (v4, v6, *(v32));
float* v33;
v33 = malloc(sizeof(float));
*(v33) = .5;
connect_dir_weighted (v7, v6, *(v33));
float* v34;
v34 = malloc(sizeof(float));
*(v34) = 333;
connect_dir_weighted (v2, v5, *(v34));
float* v35;
v35 = malloc(sizeof(float));
*(v35) = 15;
connect_dir_weighted (v4, v5, *(v35));
char ** v36;
```

```
v36 = malloc(sizeof(char *));
*(v36) = malloc(strlen("Graph Initialized\n") + 1);
strcpy(*(v36), "Graph Initialized\n");
printf("%s", *(v36));;

v11 = init_node("");
(v11)->data = "";
node_t ** v37;
v37 = &(v2);
node_t ** v38;
v38 = &(v11);
*(v38) = *(v37);
float* v39;
v39 = malloc(sizeof(float));
*(v39) = 0;
float* v40;
v40 = &(v12);
*(v40) = *(v39);
node_t ** v41;
v41 = &(v11);
entry_t*** v42;
v42 = &((*(v41))->out);
int v69;
entry_t* v43;
void* v44;
if (*(v42)) {
for (v69 = 0; v69 < TABLE_SIZE; v69 = v69 + 1) {
for (v43 = (*(v42))[v69]; v43; v43 = (v43)->next) {
v44 = (v43)->key;
char ** v70;
v70 = malloc(sizeof(char *));
*(v70) = malloc(strlen("current node: ") + 1);
strcpy(*(v70), "current node: ");
printf("%s", *(v70));
node_t ** v71;
v71 = &(v44);
printf("%s", "N-");
printf("%d", (int)(*(v71)));
printf("%s", "(\"");
printf("%s", (char *)((*(v71))->data));
printf("\")");
char ** v72;
v72 = malloc(sizeof(char *));
*(v72) = malloc(strlen("\n") + 1);
strcpy(*(v72), "\n");
printf("%s", *(v72));
char ** v73;
v73 = malloc(sizeof(char *));
*(v73) = malloc(strlen("\n") + 1);
strcpy(*(v73), "\n");
printf("%s", *(v73));;
float* v74;
v74 = &(v12);
float* v75;
v75 = malloc(sizeof(float));
*(v75) = 1;
float* v76;
v76 = malloc(sizeof(float));
*(v76) = (*(v74) + *(v75));
float* v77;
v77 = &(v12);
*(v77) = *(v76);

list_t ** v78;
v78 = &(v9);
node_t ** v79;
v79 = &(v2);
list_t ** v80;
v80 = malloc(sizeof(void));
*(v78) = node_add_back(*(v78), *(v79));;
list_t ** v81;
v81 = &(v8);
node_t ** v82;
v82 = &(v44);
list_t ** v83;
v83 = malloc(sizeof(void));
*(v81) = node_add_back(*(v81), *(v82));;
entry_t*** v84;
v84 = &(v10);
node_t ** v85;
v85 = &(v44);
float* v86;
v86 = &(v12);
node_t * v88;
v88 = *(v85);
float v87;
v87 = *(v86);
*(v84) = put_node(*(v84), (node_t *)(v88), (void*)(&(v87)));
list_t ** v89;
v89 = &(v8);
node_t ** v90;
v90 = peek(*(v89));
node_t ** v91;
v91 = &(v11);
*(v91) = *(v90);
list_t ** v92;
v92 = &(v8);
void* v93;
*(v92) = pop(*(v92));;
}
}
} else {

}
printf("{");
entry_t*** v95;
v95 = &(v10);
int v104;
```

```
entry_t* v96;
void* v97;
if (*(v95)) {
for (v104 = 0; v104 < TABLE_SIZE; v104 = v104 + 1) {
for (v96 = (*(v95))[v104]; v96; v96 = (v96)->next) {
v97 = (v96)->key;
node_t ** v105;
v105 = &(v97);
printf("%s", "N-");
printf("%d", (int)(*(v105)));
printf("%s", "(\"");
printf("%s", (char *)((*(v105))->data));
printf("\")");;
char ** v106;
v106 = malloc(sizeof(char *));
*(v106) = malloc(strlen(": ") + 1);
strcpy(*(v106), ": ");
printf("%s", *(v106));;
entry_t*** v107;
v107 = &(v10);
node_t ** v108;
v108 = &(v97);
float* v109;
v109 = (float*)(get_node(*(v107), *(v108)));
printf("%.3f", *(v109));;
char ** v110;
v110 = malloc(sizeof(char *));
*(v110) = malloc(strlen(", ") + 1);
strcpy(*(v110), ", ");
printf("%s", *(v110));;
}
}
} else {

}
printf("}");
char ** v111;
v111 = malloc(sizeof(char *));
*(v111) = malloc(strlen("\n\n") + 1);
strcpy(*(v111), "\n\n");
printf("%s", *(v111));;
}
```

Listing 39: bst.dots.c

E.4 tutorial.dots.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dict.h>
float v1;
char * v2;
node_t * v3;
node_t * v4;
graph_t * v5 = NULL;
list_t * v6 = NULL;
entry_t** v7 = NULL;
int main (int argc, char ** argv)
{
float* v8;
v8 = malloc(sizeof(float));
*(v8) = 5.3;
float* v9;
v9 = &(v1);
*(v9) = *(v8);
char ** v10;
v10 = malloc(sizeof(char *));
*(v10) = malloc(strlen("This is a d.o.t.s. program.") + 1);
strcpy(*(v10), "This is a d.o.t.s. program.");
char ** v11;
v11 = &(v2);
*(v11) = *(v10);

v3 = init_node("");
(v3)->data = "";
char ** v12;
v12 = malloc(sizeof(char *));
*(v12) = malloc(strlen("florida") + 1);
strcpy(*(v12), "florida");
v4 = init_node("");
(v4)->data = *(v12);
node_t ** v13;
v13 = &(v3);
node_t ** v14;
v14 = &(v4);
graph_t ** v15;
v15 = malloc(sizeof(graph_t *));
*(v15) = (node_plus_node(*(v13), *(v14)));
graph_t ** v16;
v16 = &(v5);
*(v16) = *(v15);
float* v17;
v17 = malloc(sizeof(float));
*(v17) = 2.3;
connect_dir_weighted (v3, v4, *(v17));
connect_dir(v4, v3);
list_t ** v18;
v18 = malloc(sizeof(list_t *));
*(v18) = NULL;
list_t ** v21;
node_t ** v19;
v19 = &(v3);
*(v18) = node_add_back(*(v18), v19);
node_t ** v20;
v20 = &(v4);
*(v18) = node_add_back(*(v18), v20);
v21 = v18;
```

```
list_t ** v22;
v22 = &(v6);
*(v22) = *(v21);
entry_t*** v23;
v23 = &(v7);
char ** v24;
v24 = malloc(sizeof(char *));
*(v24) = malloc(strlen("e") + 1);
strcpy(*(v24), "e");
float* v25;
v25 = malloc(sizeof(float));
*(v25) = 12.7;
float v26;
v26 = *(v25);
*(v23) = put_string(*(v23), *(v24), (void*)(&(v26)));
entry_t*** v27;
v27 = &(v7);
char ** v28;
v28 = malloc(sizeof(char *));
*(v28) = malloc(strlen("a") + 1);
strcpy(*(v28), "a");
float* v29;
v29 = malloc(sizeof(float));
*(v29) = 8.17;
float v30;
v30 = *(v29);
*(v27) = put_string(*(v27), *(v28), (void*)(&(v30)));
entry_t*** v31;
v31 = &(v7);
char ** v32;
v32 = malloc(sizeof(char *));
*(v32) = malloc(strlen("d") + 1);
strcpy(*(v32), "d");
float* v33;
v33 = malloc(sizeof(float));
*(v33) = 4.25;
float v34;
v34 = *(v33);
*(v31) = put_string(*(v31), *(v32), (void*)(&(v34)));
float* v35;
v35 = &(v1);
printf("%.3f", *(v35));
char ** v36;
v36 = malloc(sizeof(char *));
*(v36) = malloc(strlen("\n") + 1);
strcpy(*(v36), "\n");
printf("%s", *(v36));
char ** v37;
v37 = &(v2);
printf("%s", *(v37));
char ** v38;
v38 = malloc(sizeof(char *));
*(v38) = malloc(strlen("\n") + 1);
strcpy(*(v38), "\n");
```

```c
printf("%s", *(v38));
node_t ** v39;
v39 = &(v3);
printf("%s", "N-");
printf("%d", (int)(*(v39)));
printf("%s", "(\"");
printf("%s", (char *)((*(v39))->data));
printf("\")");
char ** v40;
v40 = malloc(sizeof(char *));
*(v40) = malloc(strlen("\n") + 1);
strcpy(*(v40), "\n");
printf("%s", *(v40));
node_t ** v41;
v41 = &(v4);
printf("%s", "N-");
printf("%d", (int)(*(v41)));
printf("%s", "(\"");
printf("%s", (char *)((*(v41))->data));
printf("\")");
char ** v42;
v42 = malloc(sizeof(char *));
*(v42) = malloc(strlen("\n") + 1);
strcpy(*(v42), "\n");
printf("%s", *(v42));;
graph_t ** v43;
v43 = &(v5);
node_t * v45;
list_t * v44 = NULL;
for (v44 = (*v43)->nodes; v44; v44 = (v44)->next) {
v45 = (node_t *)((v44)->data);
node_t ** v46;
v46 = &(v45);
printf("%s", "N-");
printf("%d", (int)(*(v46)));
printf("%s", "(\"");
printf("%s", (char *)((*(v46))->data));
printf("\")");
char ** v47;
v47 = malloc(sizeof(char *));
*(v47) = malloc(strlen("\n") + 1);
strcpy(*(v47), "\n");
printf("%s", *(v47));;
}
printf("[");
list_t ** v49;
v49 = &(v6);
node_t * v51;
list_t * v50 = NULL;
for (v50 = *(v49); v50; v50 = (v50)->next) {
v51 = *((node_t **)((v50)->data));
node_t ** v52;
v52 = &(v51);
printf("%s", "N-");
printf("%d", (int)(*(v52)));
printf("%s", "(\"");
printf("%s", (char *)((*(v52))->data));
printf("\")");;
char ** v53;
v53 = malloc(sizeof(char *));
*(v53) = malloc(strlen(", ") + 1);
strcpy(*(v53), ", ");
printf("%s", *(v53));;
}
printf("]");
char ** v54;
v54 = malloc(sizeof(char *));
*(v54) = malloc(strlen("\n") + 1);
strcpy(*(v54), "\n");
printf("%s", *(v54));
printf("{");
entry_t*** v56;
v56 = &(v7);
int v65;
entry_t* v57;
void* v58;
if (*(v56)) {
for (v65 = 0; v65 < TABLE_SIZE; v65 = v65 + 1) {
for (v57 = (*(v56))[v65]; v57; v57 = (v57)->next) {
v58 = (v57)->key;
char ** v66;
v66 = &(v58);
printf("%s", *(v66));;
char ** v67;
v67 = malloc(sizeof(char *));
*(v67) = malloc(strlen(": ") + 1);
strcpy(*(v67), ": ");
printf("%s", *(v67));;
entry_t*** v68;
v68 = &(v7);
char ** v69;
v69 = &(v58);
float* v70;
v70 = (float*)(get_string(*(v68), *(v69)));
printf("%.3f", *(v70));;
char ** v71;
v71 = malloc(sizeof(char *));
*(v71) = malloc(strlen(", ") + 1);
strcpy(*(v71), ", ");
printf("%s", *(v71));;
}
}
} else {

}
printf("}");
char ** v72;
v72 = malloc(sizeof(char *));
```

```
*(v72) = malloc(strlen("\n") + 1);
strcpy(*(v72), "\n");
printf("%s", *(v72));;
}
```

Listing 40: tutorial.dots.c

F Git Commit History

```
commit 9a32082c6edab4f17fcdc72c34350de433b91cc4
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 22 11:22:46 2015 -0500

    cleaned up code

commit 5eb8a0527a529fbaea3cfcacbb0eefd7177ffa52
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 22 10:58:39 2015 -0500

    got rid of unused rules by having fdecls use the alternate stmt rule that
        doesn't include fdecl (no nested fdecls)

commit 4bf1c9acced90a35bd1607b6a2638f45c4b6502c
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 22 10:50:00 2015 -0500

    cleaned up runtest.py

commit 154e9dad2d9f7da88caad984a68e73ce678662e6
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 22 01:22:55 2015 -0500

    removed commented out code. added clean to the setup reule in Makefile

commit 78869c4034340a6aeba668c523863bd58262897d
Merge: 12b7090 8bd68d1
Author: Adam Incera <aji2112@columbia.edu>
Date: Tue Dec 22 01:00:46 2015 -0500

    merge

commit 12b7090680d61ae64d2c2941fc7d2e9e47ccf416
Author: Adam Incera <aji2112@columbia.edu>
Date: Tue Dec 22 00:59:58 2015 -0500

    cleaning out src directory

commit f5c9d3d34302c5b9246c0a79419b2f6065484e04
Author: Adam Incera <aji2112@columbia.edu>
Date: Tue Dec 22 00:55:07 2015 -0500

    cleaned up translator and analyzer

commit 8bd68d157e51c8113e1142142fd4bfba58f0bf4e
```

```
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 22 00:23:34 2015 -0500

   added some image files for our report. removed extraneous file from src

commit ff9b8a16d22bb9746428d21ed4671f588f8bbdf4
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 19:58:05 2015 -0500

   put copies of our example code in src/sample-code to make it easier for TAs
        to find

commit aed9f26d63082bf69f78afac27cc36248110fbeb
Merge: 4b89c4b e4008a6
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 19:53:46 2015 -0500

   Merge branch ’compile’

   merged current compile branch into master

commit e4008a6284d765a2b50e777e83e0e7721f9682f0
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 19:53:20 2015 -0500

   minor changes to make output prettier

commit 4b89c4be9e2fde92ab5dfdd4bd3ec5960090d6df
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 09:12:02 2015 -0500

   text

commit 31315eed5b12031f7676c19a00d8d4a53bc566c3
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 09:11:17 2015 -0500

   format

commit 5cb160fe68a92a85e81f240effe6a13893b960b1
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 08:56:00 2015 -0500

   syntax highlighting

commit 8c373b266992ceddf06e472d22f5f6f82f65d00a
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 08:53:34 2015 -0500

   syntax high lighting

commit 19f843182137359f2c74f881e32d22a90afaa0f8
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 08:49:14 2015 -0500
```

```

    hosanna

commit 2d91001030f8c7894832725679f337621fafbc57
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 08:48:04 2015 -0500

    format changes

commit 170dfcf384ec1d664e3a5b0c92adf224d5e3f290
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 08:47:06 2015 -0500

    compilation in structions

commit f04a58b268747796d6cee19e412c736ac2035da1
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 08:41:39 2015 -0500

    duplicate readme

commit e3d9cc6f6ee456521bb80de8f199745359d42ae8
Author: Hosanna Fuller <Miramonte23@gmail.com>
Date: Mon Dec 21 08:41:13 2015 -0500

    update to readme.md

commit 0324e8834ffc893f5a4ce97c22b3aade004d68fe
Merge: 4ab2577 10d6dd8
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 06:06:50 2015 -0500

    mergeMerge branch 'compile' of https://github.com/adamincera/dots into
       compile

commit 4ab2577fd7105c852adbfe57aa9b6bc136d6f95e
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 06:06:48 2015 -0500

    sample files

commit 1cccafbb5739cf8999458ac8cad837461cf664cf
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 06:04:46 2015 -0500

    sample algors

commit 10d6dd880f532a8845a08a2b77a82625ff6e3201
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 05:43:49 2015 -0500

    working bst

commit 6b9ce3d237c0a804e63dc665a8eac2becb9391cc
```

```
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 05:13:44 2015 -0500

   pushing

commit 95012f41c8dd6fbcd6009a0238b5483721c9522e
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 03:51:19 2015 -0500

   fixes for the segfault on values

commit 6a5bd4d61d1bb78c23ecf25030e5bd571f4fad98
Merge: 72ea472 51954bb
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 02:36:10 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile

commit 72ea47227eacf44d9d89d0e52ba255a35ce7ba4c
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 02:36:00 2015 -0500

   temp

commit 51954bb6e94fff2185f5df86ce8dd8401bb48c4a
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 21 02:35:25 2015 -0500

   took out free

commit 28ed5d5ba4cd10c20dd8054e86b80e04c41f9f56
Merge: 9e7d310 31c2276
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 21 02:06:17 2015 -0500

   merge

commit 9e7d3101cc7f6df5a13d4be36b27e1587aad460c
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 21 02:06:08 2015 -0500

   graph + node

commit 31c2276f761cc4fd9afeeabc8c9a8457b7a797d7
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 01:53:09 2015 -0500

   adds

commit fdcbc66caa0a13dcd68b19afb77707b2be9171e0
Merge: eea7a53 791c1c4
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 01:11:21 2015 -0500

```

```
    Merge branch 'compile' of github.com:adamincera/dots into compile

    merge

commit eea7a53717a4ad32398b707b43cd9d7afc756f4b
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 01:11:04 2015 -0500

    fixed bfs error. altered if stmt handling in analyzer.ml

commit 791c1c406a5b771d9f1bf8c4e97467baf359d572
Merge: 5b462e1 67a35d2
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 21 01:03:38 2015 -0500

    merge

commit 5b462e119334f4494063df1d9a865a0685046cb6
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 21 01:03:30 2015 -0500

    pulling

commit 67a35d2d96d5f08715a706bcd471fe8965697a2d
Merge: 88995a8 df268e8
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 00:39:04 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit 88995a86b22cc34353fd99c3377496a7d3fb493d
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 21 00:38:49 2015 -0500

    alterations

commit df268e835a18e20963be66daa9b995875c0fa03e
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 00:31:36 2015 -0500

    rcg and hosanna added breadth first search example

commit 5c51b503c5d694af5d448e3e419e73d7f1b4f5dc
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 00:29:59 2015 -0500

    test cases

commit 4b53058ddc80d55cdfb9240ab0c8608b613ed813
Merge: 705c5ae ef256a2
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 00:27:27 2015 -0500
```

```

    Merge branch 'compile' of github.com:adamincera/dots into compile

    merge

commit 705c5aef630cbda9b5e731a8b6e03d7d591e9360
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 21 00:27:10 2015 -0500

    rachel & hosanna fixed enqueue call

commit 241eff4d6ab8d73bf27c7bed5ed0d5f132055ad5
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 23:48:09 2015 -0500

    Test case

commit ef256a2dcadcf54609620fd2e6a931c5768bee09
Merge: 2a4799f 84cdfcc
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 23:32:33 2015 -0500

    merge

commit 2a4799f32cfb6373200ba89f66863d813b6f7b7a
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 23:32:08 2015 -0500

    pulling

commit e73a862e8e193c77ea0bb656792d93412396a973
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 23:31:24 2015 -0500

    added initialization into graph_plus_node()

commit 84cdfcceda948f345ce831a5ef3ed892afac8ca4
Merge: a706f7a fcc869e
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 23:27:09 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit a706f7af90e14bcb7888b27185db9e849bb618da
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 23:26:49 2015 -0500

    fixed node + node

commit fcc869ed89263b5a581c5465a510455ddf48b492
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 23:23:59 2015 -0500
```

```

    simplified dijkstra

commit 02fdef29963dc95b880a803562300346a14290cb
Merge: 83f7c1a 9be880b
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 23:12:47 2015 -0500

    mergin

commit 83f7c1aef9741b6f7c4b6a2295c0e810417508bc
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 23:12:18 2015 -0500

    test for initializing graph in add_node

commit e3101e2ca2cbdeed3fd9089d6935db0352d36b69
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 23:10:12 2015 -0500

    pulling

commit 4e2321b2d7f9cf7714fa871317ab0e93b578d0d3
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 23:09:46 2015 -0500

    integrated init_graph() into add_node

commit 9be880b6e2fef401c493e7f175963989920e125b
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 23:03:19 2015 -0500

    fleshed out test case for ine and oute

commit 099c99b9c483fe5cfef5c479267e9f15ec78a859
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 22:55:49 2015 -0500

    fixed oute

commit f8e5bc6865ef92f974a2bf69b831d2e9f930f51b
Merge: 8b0f6ec 90d7f7a
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 22:08:08 2015 -0500

    Merge branch 'compile' of github.com:adamincera/dots into compile

commit 8b0f6ecdf831c11b473684a00de355e229fa8598
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 22:08:02 2015 -0500

    progress

commit 90d7f7ae1613ce5a2a1e7a66d0bfdb1818a616e1
```

```
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 22:03:47 2015 -0500

    fixed dict removal

commit 25078894c0fe748dd25635197783lb062958d6ef
Merge: db88290 6ef44bb
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 21:41:09 2015 -0500

    merge

commit db88290ae4748a9c653e4d7ceb9151eff2861d72
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 21:41:00 2015 -0500

    pulling again

commit 6ef44bbbc8c554e15d9d59b7c344a6705fd6f402
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 21:38:27 2015 -0500

    fixed the printing of random % fmt strings

commit 58f36f41fece18aaf675cde201abbc8660e5c34c
Merge: 65adba5 9dc3ce2
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 21:23:16 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit 65adba5ff62c2456cd8d9939eaafbf3a8db68075
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 21:23:15 2015 -0500

    partial dijkstras

commit 9dc3ce2c5d95d27db9ae37e916c9bcc3cc62de83
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 21:21:02 2015 -0500

    all the dots tests pushed

commit 44c60b245827fb250f833c4e210abc6beff488f3
Merge: cd03369 a4cc926
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 21:13:48 2015 -0500

    merge

commit cd0336936b5253f8c3c22c0d6a56a51be73e393c
Author: Adam Incera <aji2112@columbia.edu>
```

```
Date: Sun Dec 20 21:13:32 2015 -0500

   pulling rachoho

commit a4cc9269a942b684c1c50e6df40ceb7720f0d605
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 21:12:59 2015 -0500

   remove

commit 1a703e81a775bc1c07cf3f29ef40c0687e52a640
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 21:12:28 2015 -0500

   ll

commit a67f79e5da30ca9698a1745313eedfeff221f6e6
Merge: 4233073 69c734c
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 20:35:49 2015 -0500

   pulling double pointer fix

commit 69c734c29fcedf721b256454f5caa15fae14c63c
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 20:35:18 2015 -0500

   fixes to graph_t declarations. min/max handling

commit 4233073102750450d5bb9cf816c942ba36bfd528
Merge: b6b6b9a bf565c1
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 20:27:56 2015 -0500

   pulled

commit b6b6b9ada1b889d8b7778a79ac72639e67303ed9
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 20:27:43 2015 -0500

   pulling

commit f5747f4e7da7c1e4a9929979d0239d631cce0093
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 19:59:53 2015 -0500

   pulling

commit bf565c1d7942792142eabe736d874b167217e379
Merge: 13361e6 d6cbe16
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 19:01:46 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile
```

```

commit 13361e61fbbf66a2c322514efa9c32153e61d16c
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 19:01:30 2015 -0500

   print for node fixed

commit d6cbe167bb7c6977e395681b7fb7bc8181c21400
Merge: 284e2c2 73d4116
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 18:17:25 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

   merge

commit 284e2c23e9d7eaf3289bee478245513f4be361d9
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 18:17:15 2015 -0500

   fixed while loop condition handling

commit 73d411672a58631cd732538115a4eaa4cbc5d26d
Merge: a2460fd 07a6a1c
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 18:04:47 2015 -0500

   fixed merge conflict

commit a2460fd848e4a4f981fec65e6e6698dae7083a08
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 18:03:01 2015 -0500

   fancy graph assignment, fixing other stuff

commit 07a6a1c5c82d1f150774e9322428fef0d69d2c04
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 16:43:21 2015 -0500

   fixes to while loop

commit 865461a8da1968319b11ab30937d9d49e0d150dc
Merge: c860bcf ed05ff9
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 16:25:47 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

   merge

commit c860bcf73d07e1ff68dc321d3698859bd290ebba
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 16:25:41 2015 -0500

```

```
    initial work on making while loops conform to the reference variable
       standard

commit ed05ff936228e4915f64e2e9feb6cd491916acf9
Merge: e7f6ecd 8d099f2
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 16:13:28 2015 -0500

    mergin

commit e7f6ecd98d222bc6bc0f45c5ac146f1de06e48e4
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 16:12:57 2015 -0500

    dict printing, fixed some C bugs

commit a9f74572627467da5896c9b20f528de5634ad67d
Merge: e2c76ee 8d099f2
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 13:55:38 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merg

commit 8d099f23c7f63e32abe657af12dd359a057c2caa
Merge: f81bf59 e493890
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 13:55:19 2015 -0500

    Merge branch 'compile' of github.com:adamincera/dots into compile

commit f81bf5908eb8c4c6854f7a26dae4cb0330328982
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 13:55:09 2015 -0500

    fully tested len built in function

commit e2c76ee717353dc4acfe3d65229b3fbbda2ecf62
Merge: f0301b4 e493890
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 13:19:28 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit f0301b4664e506a244cf21979b9c515127fd82bd
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 13:19:22 2015 -0500

    moved call rule into the term rule

commit e493890422a6c12fdf88400b44b51a9eb4004c59
```

```
Merge: e81fc67 1d469e7
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sun Dec 20 13:16:27 2015 -0500

   Merge branch ’compile’ of https://github.com/adamincera/dots into compile

commit e81fc67922be63cdd1687dd640a5b26715ad587c
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sun Dec 20 13:16:10 2015 -0500

   fixed function args not taking in types problem

commit 05f5d928d69e486b407bc8f8db06740d16398a0f
Merge: a769efc 1d469e7
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 12:51:32 2015 -0500

   Merge branch ’compile’ of https://github.com/adamincera/dots into compile

   merge

commit a769efcbf69b5722b6b9c1817aecb053576d2664
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 12:51:30 2015 -0500

   minor change

commit 1d469e7be1d99a34f134d30ad702572e75049f5e
Merge: 235e792 e94b8cd
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 12:46:03 2015 -0500

   hoho and ratchel and adamame to the rescue

commit 235e7929cb7b3ab9004679c6dcc276a0cca4f1da
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 12:43:50 2015 -0500

   progress

commit e94b8cd2e90d17521c7ecd6918ad04be746409ac
Merge: 3ec97b9 2d60a71
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 11:26:57 2015 -0500

   Merge branch ’compile’ of https://github.com/adamincera/dots into compile

   merge

commit 3ec97b973f2ed99f7b68d76ef35ef78c24e4bb6b
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 11:26:39 2015 -0500

```

```
    implemented c translation for generic function call. i.e. implemented
       handling of dealing with result vars

commit 2d60a71f17b0c4a206fc686c10f27e93cd289bd1
Merge: 1f09692 56edcb2
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Dec 20 11:08:48 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit 1f09692c505abba250a022e78cee6b4379409598
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Dec 20 11:08:21 2015 -0500

    gotta print stderr too... in case of segfault yayaya

commit 56edcb2bdd8c6e3e30da00769330b3e873ee375a
Merge: 160aa51 1a2112f
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 11:01:02 2015 -0500

    merge conflicts for handled

commit 160aa51ee47faad64be92a4aebe0433412d2613f
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 10:58:42 2015 -0500

    all member call functions working

commit 1a2112fdbbe3cb4ea579080942417bab83109094
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 10:11:36 2015 -0500

    made fixes to AccessAssign. it looks like maybe the underlying C function
       is broken

commit 0f82e8be1bd8d2595127fc27124d3b07630ec988
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 09:52:05 2015 -0500

    fixed list access translation

commit 04f82b6d6b68598ced75040c953979360d161db6
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 09:39:26 2015 -0500

    fixed list printing

commit da3b51f9fbdfa0665c488d640ef3174cbf398f90
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 08:26:58 2015 -0500

    for dict progress

```

```
commit fcbf505f9efc0570cd323cd3a7f1e538700b1136
Merge: 41dae0d 8ebd9f0
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 06:45:59 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit 8ebd9f023c68b6d6bcb22b7858556b4591443a43
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 06:45:47 2015 -0500

    put init_dict() into put_whatever

commit 41dae0de4b8546337d42082b5e1e1d45d08097e3
Merge: b6613da 8a6401c
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 06:45:09 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit b6613da54f020c9f7d82564607c07bfcc40906fa
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 06:45:06 2015 -0500

    altered Dict declaration to initialize it to NULL. more work on for dict

commit 1fc17dda95c9758cf0fc549791b98111fda62573
Merge: 6883e45 8a6401c
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 06:25:28 2015 -0500

    Merge branch 'compile' of github.com:adamincera/dots into compile

commit 6883e458cdf0a6665a31b1934a02fc8b52dfaad9
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 06:25:21 2015 -0500

    out member f in progress

commit 8a6401c5d02064248edfb8c1ad41cf240f721158
Merge: e7fd4b1 fbbacd3
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 06:22:57 2015 -0500

    pulled pork

commit e7fd4b101010146d7c684068be8192b76bbbebba
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 06:22:28 2015 -0500

```

```
    implemented edgeops! ~meaningful 6am commit message~

commit fbbacd3f43bf3c9eb2b8c64e81699e089956accf
Merge: 1214dda e588611
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 05:54:59 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit 1214ddacf8206a801ebf958d33f01722d7b63dcb
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 05:54:53 2015 -0500

    fixing For loop translation over dict

commit e588611356c7e187fe99a25dbe4c0312a8848a75
Merge: 90c7b04 df7e52f
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 05:53:59 2015 -0500

    to: ratchel from: adamame

commit 90c7b04d1f3df63ddb5d8dd036dc0502f3338702
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 05:47:30 2015 -0500

    sending things to ratchel

commit df7e52f1a695d7f3c5648c334e015f1152a647c1
Merge: e085ef5 b8d46bb
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 05:44:18 2015 -0500

    Merge branch 'compile' of github.com:adamincera/dots into compile

commit e085ef5958acabc963c4b9806dcf42b3bf7bfb64
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 05:44:04 2015 -0500

    pop and enqueue implemented and tested

commit ab376d38b61c3b1ada36f3076a00256433d66ac5
Merge: 98c6f89 b8d46bb
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 02:46:48 2015 -0500

    pulled

commit 98c6f89dba91ea866d8dde397412084089c7b33e
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 20 02:46:23 2015 -0500

```

```
    tryna pull

commit b8d46bb4ae5e93c70cba74dbc2defc5fa01bcf05
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 01:48:35 2015 -0500

    fixed list literal translation

commit cc2e019aa2216ca68b8e607456717b8153c86bb4
Merge: a9b78cc 3f8550b
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 01:38:32 2015 -0500

    Merge branch ’compile’ of github.com:adamincera/dots into compile

commit a9b78cc4750cd930ecafd96cdf25d77df7942a64
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 01:38:23 2015 -0500

    progress

commit 3f8550b283135b9b3ed874cdffbce0e10db90bc1
Merge: 9632c38 e561b42
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 01:37:57 2015 -0500

    Merge branch ’compile’ of https://github.com/adamincera/dots into compile

    merge

commit 9632c38ce43fe3a97507161fb8bdac3add94b28e
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 01:37:54 2015 -0500

    improvements to implementation of list literal

commit e561b4200c73246d33236525a2c2fd85bffff5ed
Merge: a793e08 bd4b52d
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 01:18:13 2015 -0500

    Merge branch ’compile’ of github.com:adamincera/dots into compile

commit a793e08c4e462d89d32378c5249cb2d7800a297b
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 01:18:00 2015 -0500

    binop working for ints, longs, and floats

commit bd4b52d0a8f7192c529346e32bff0f49f6898b02
Merge: a33cdbf ea84af7
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 00:24:13 2015 -0500

```

```
    Merge branch ’compile’ of https://github.com/adamincera/dots into compile

    merge

commit a33cdbfcf712ec6ff104aa9f670e472f9ab56664
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 20 00:24:02 2015 -0500

    implemented list literal translation. still needs to be tested

commit ea84af7421f25ebb4cca048c839ad66e3fab196c
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 20 00:14:07 2015 -0500

    added the strliteral and numliteral to add auto var return

commit 1ef7378189f7c538b212eb5cb64a9e58091de457
Merge: f5883a8 eecfae5
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 23:35:18 2015 -0500

    pulled

commit f5883a848ffc187d0dc5da618946d4050e72a4b9
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 23:35:00 2015 -0500

    tryna pull

commit eecfae5f97be97343e39c7663ea5d22431929cbf
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 22:31:46 2015 -0500

    added printf that works

commit 04560b6f4c49ae87a8e95c991f60fd65a70291a7
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 22:05:08 2015 -0500

    altered Access translation to deal with pointers

commit 95232deef2443bd398c9c5e03c580ec6062bc33f
Merge: d1eff43 5e3b9d7
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 21:49:30 2015 -0500

    Merge branch ’compile’ of github.com:adamincera/dots into compile

    g

commit d1eff431c4f45e7cb13fb9d9651e8b14c49c9b93
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 21:49:14 2015 -0500

```

```
    Changed handling of Id, NumLiteral to work with pointers for result vars

commit 5e3b9d73500a515dc14c44fe655ecab91278112e
Merge: d8872c3 f3cc977
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 21:27:10 2015 -0500

    merging

commit d8872c3719212c7663e966e8a813d1056307c4d5
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 21:25:04 2015 -0500

    added len to analyzer

commit f3cc977205d01e00ec7073fe6e74591420280d39
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 21:11:39 2015 -0500

    added comment explanation for how to do automatic result variable handling.
        implemented Access with result_var handling

commit 5e45c5a5ed585a29e32f0facfa1d65f19ac69f3e
Merge: f50676f a7e3012
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 20:40:30 2015 -0500

    merge handled

commit f50676f1026d14e2d016bfc4526c2d060e0f22e4
Merge: 7e7b70d f40e18a
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 20:38:49 2015 -0500

    altered function names for sexpr_type

commit a7e30128936e2fb244fed5372219f968787ab86d
Merge: eabe61b 341af15
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 20:28:34 2015 -0500

    pulling

commit eabe61b26c43c729ad80fe52ab2268f6be3d96ee
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 20:28:15 2015 -0500

    compiling version of let function

commit 341af15e60a0eef1636db0ccca852f6ba3b04df5
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Dec 19 20:19:57 2015 -0500

    tried to fix it
```

```

commit 4e1856e072accb6056760e8a9111250e4b1f01be
Merge: 5adcae3 277f28f
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 19 19:24:29 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit 5adcae380eb8553f4c7ffbf25cfb00938701bd67
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 19 19:24:15 2015 -0500

   analyzer accessassign

commit 277f28f12fad566351b4669d16ff0abcf55f3eaf
Merge: 466f003 f40e18a
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 18:57:21 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

   merge

commit 466f00306d2170d47708e38a4fab26293f51d002
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 18:57:12 2015 -0500

   fixed For loop declaration of auto var

commit 7e7b70d343737b97b84e29b9940d7ab282d4cbe3
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 18:15:40 2015 -0500

   prgress

commit f40e18a56ef3c56cc01c0b09a81901ac3a350164
Merge: f371a5e 7013610
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 18:11:38 2015 -0500

   pulling

commit f371a5e7ab65420c306bf05729e78992d364e316
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 18:11:24 2015 -0500

   added tests for len functions

commit 70136107c0e8f271430cd1e9ab76e2144acfa56b
Merge: b89f332 3b1a8c1
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 18:10:05 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile
```

```

    merge

commit b89f3326d4164c3d9b929f1eaf471401f5f96d2a
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 18:09:55 2015 -0500

    implemented translation of custom function definitions. includes: updating
       the environment variables to include names of functions and local scope
        to add parameters to variable mappings

commit 3b1a8c1f27fe3a8516ed10121cdfb1c86ceb3c92
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 18:08:19 2015 -0500

    changed nodelist_t to regular old list_t

commit 4b4dffc02af854d22d747bf1ff4c1e2a7c92edbf
Merge: 6ddeab2 4208526
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 17:54:17 2015 -0500

    pulling

commit 6ddeab22c40c3629715dae31cd1dc96588c78bc2
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 17:53:52 2015 -0500

    redefined node->in and out as dict<node, num>

commit 4208526b73136e528e0c00ffd9d53ad0027ba172
Merge: 3b27467 e9088b5
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Dec 19 17:40:23 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit 3b27467513345eff99f4bcaac423d6842d8a40e9
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Dec 19 17:40:06 2015 -0500

    typo in analyzer, wrote access assign simple test

commit e9088b5cb869d9b848257b60d4262023a9e714ba
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 17:36:11 2015 -0500

    snippet for list access assignment

commit 65462b8ba7c881b7dc1170d7b8dbd5977e3bc78e
Merge: 0537c03 e4a73d7
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 17:04:34 2015 -0500

```

```
    pulling

commit 0537c0396d902278efbaa76b570826ab7edc65ca
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 17:04:22 2015 -0500

    access and accessAssign

commit e4a73d780e1f539aac90701754ea20910159e38a
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 15:57:45 2015 -0500

    dequeue

commit 66f7fd1f2883aad57e5895fe0ddc55fdb324173e
Merge: d5a9b1f c8f68d1
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 15:35:15 2015 -0500

    Merge branch 'compile' of github.com:adamincera/dots into compile

commit d5a9b1f4f50a60bd47a606ef04a625334e7e7921
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 15:35:08 2015 -0500

    preliminary dequeue

commit c8f68d191159cdaa8567bed6f97e5507430a3762
Merge: 27dbd80 aad6167
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 15:14:26 2015 -0500

    pulleen

commit 27dbd80db0edca2295a55ca7351b33e79cf00732
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 15:11:42 2015 -0500

    fixed pop

commit aad6167ee56d4b0092a4b0ac09e95d32b5ef8674
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 14:49:17 2015 -0500

    revert list of list impossible

commit 42693e07d5f981d0feca7c838f04518ddfa986fe
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 14:47:18 2015 -0500

    added enqueue dequeue for all types

commit 9320b9b55abe1add54aef69daf81770f0423971a
Author: hosannajfull <miramonte23@gmail.com>
```

```
Date: Sat Dec 19 14:39:20 2015 -0500

   add back add front

commit 0a29c22fc86d26bbc336906ed50b3bc86de9e31c
Merge: d87d96f 5ec5a53
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 12:51:05 2015 -0500

   merge conflicts handled

commit d87d96f0a2a49a689c21731034733fd25eaf7cd4
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 12:48:29 2015 -0500

   added MemberCall for enqueue in Analyzer

commit 5ec5a5349989cd4034a012d2a883784d9d16541e
Merge: dcc692a a8627f5
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 12:47:06 2015 -0500

   pulling

commit dcc692a10bfc2d77545c78f7e2d9e5aa6fbab4bb
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 12:46:48 2015 -0500

   added min and max for dicts and lists and tests and snippets

commit a8627f565d7d331f7155c72fff6358dd8186d7ec
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 12:29:30 2015 -0500

   removed references to MemberVar

commit c9015f18b637c11471d4aca5a97577218ffbe4ed
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 12:01:32 2015 -0500

   moved old memberVar handling into memberCall. adjusted memberCall

commit d8cb810123e4a9aaf436780e24d44754e10f128f
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 11:00:08 2015 -0500

   fixed parsing rules so that dijkstra's example now compiles. and no shift/
       reduce conflicts

commit 74dc455d12d305d21f661b79dcb61ecabe6201e4
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 19 05:06:30 2015 -0500

   ok doing better... uncommented assign
```

```

commit b5d9e045dd4835d7a0f9c823ed923a395bf6a60e
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 19 04:58:37 2015 -0500

   separated logical exprs

commit a75b7a7665cdacb4dc786812c8cc18c3a0e4c854
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 19 04:50:40 2015 -0500

   fixes

commit f53f3f1961b1b7cfd4e24fcd0b4253262fe07e30
Merge: 72f7c7a f0098c6
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 03:06:57 2015 -0500

   pulling

commit 72f7c7a402b104a26d3a2ebb6e03e06dac4e8eca
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 03:06:45 2015 -0500

   added dict remove and snippets for adding nodes and graphs

commit f0098c68235cf4406cf76bbe040119d4bf893d45
Merge: 8a12b32 d3d684e
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 03:02:25 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

   merge

commit 8a12b323c0fa4d193f442a54461ffaef38ae184d
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 19 03:02:16 2015 -0500

   old versions of expr parse rule added in comments. additional test files.
      some menhir test cases

commit d3d684e89bd97728138d5424a5edfbb2be83e741
Merge: b114ceb 5b60c2e
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 02:01:22 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile

commit b114ceb92722b48c99a585a420f610c298823b01
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 19 02:01:07 2015 -0500

   added the binop add string cases and LEQ for strings
```

```

commit 0eab6f18a1bf3af3668e6aefb2cd3d591955f374
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 01:52:12 2015 -0500

   node plus node, graph plus node

commit 5b60c2eb2860ed48a4ca5435ca7fc84bc758a9d6
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 19 00:58:48 2015 -0500

   modified graph subtraction and fixed list concatenation

commit bebccacad0eaac2f48e913143feae44bf394bab8
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 23:45:28 2015 -0500

   mango and I fixed the conversion of For loops to string * expr

commit 38229fd51277f91a81043cff9f445e13438e75d4
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Fri Dec 18 22:38:08 2015 -0500

   have to add the node case

commit ce5a2232279f14689b1e5463da0c75ac3446cdf4
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Fri Dec 18 22:06:08 2015 -0500

   first attempt at fixing for loops to do 'for x in expr'

commit 7f2c428b09230fcdfc8a5011c4ccd4cd63447183
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 20:42:01 2015 -0500

   fixed dict printing

commit 7d2cc33083eb5e5ec1322e916a44eecd6840d6fc
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 20:33:08 2015 -0500

   fixed library includes

commit eda2688ce9d5f6a9f53e861142f209a42e02a07a
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 20:31:53 2015 -0500

   actually fixed merge conflict

commit 553be92fa6ef346c51e8b4b08ba1dc39687ceb9b
Merge: dc20867 9368ba3
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 20:29:18 2015 -0500

```

```
    addressed merge conflict

commit dc20867f5b3db6135abe6f7226d1416694f5388f
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 20:28:11 2015 -0500

    implemented dict printing

commit 9368ba3fd266f03b0e466f043dbb3219a32cdaf5
Author: Adam Incera <aji2112@columbia.edu>
Date: Fri Dec 18 20:27:59 2015 -0500

    it compiles now

commit d9c6dc28398cdbe536de16c4fd0ccc38d592d923
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 20:24:41 2015 -0500

    pusheen

commit 2a82bcc6767e84204c5ca47e8842f4ad7f9c3c0d
Merge: f8da6b5 5d1c173
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 19:01:18 2015 -0500

    Merge branch 'compile' of github.com:adamincera/dots into compile

commit f8da6b5a0eb61f4b966f20ecf9a8885f5854dea0
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 19:01:02 2015 -0500

    fixed it

commit 5d1c173f9f8bb80eb6ed534bd0871c3386404aaa
Author: Adam Incera <aji2112@columbia.edu>
Date: Fri Dec 18 18:57:34 2015 -0500

    fixed for node in graph snippet

commit 2b3e8af41f3c75152b44f83a4e544c87e3e68b6d
Author: Adam Incera <aji2112@columbia.edu>
Date: Fri Dec 18 18:42:15 2015 -0500

    list access and corresponding test

commit 20e843f0ac4f11a93ce94c731e796e43de5996dd
Merge: 3af0cd9 677b93f
Author: Adam Incera <aji2112@columbia.edu>
Date: Fri Dec 18 18:31:05 2015 -0500

    pulling once again

commit 3af0cd9c9e0752d9f3806071bb4c2df931ca96ab
Author: Adam Incera <aji2112@columbia.edu>
```

```
Date: Fri Dec 18 18:30:37 2015 -0500

   generalized list insertion/copying and added corresponding tests

commit 677b93fb5889e409be466fdd57f0a7a6ead98289
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Fri Dec 18 18:28:48 2015 -0500

   fixed some missing pattern matches, bug with nostmt

commit 8c007cf5af7e89c5eadd6883b9e86df9eb6f5127
Merge: 5520037 19e32cf
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Fri Dec 18 18:20:50 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit 5520037c35e4c065084cd6629782ca3c2ff1f8c6
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Fri Dec 18 18:20:39 2015 -0500

   AccessAssign first attempt

commit 19e32cf5fdb7d7a9d0161987a4107127503a2832
Merge: 7c86f44 efa7018
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 17:52:58 2015 -0500

   merge conflicts handled

commit 7c86f44b0d363730e8619c7345a8e6ba003b05fe
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 17:51:47 2015 -0500

   string of stmt

commit efa70180554b314eef0c75e0035b707aba6675a0
Author: Adam Incera <aji2112@columbia.edu>
Date: Fri Dec 18 16:51:50 2015 -0500

   snippets for peek and pop

commit 7d1f7d08195b8038ee07da074b2febce69edcaae
Merge: 06348ce 63eb56b
Author: Adam Incera <aji2112@columbia.edu>
Date: Fri Dec 18 16:46:43 2015 -0500

   merging

commit 06348cee49562728265008e4b74a711af585a42d
Author: Adam Incera <aji2112@columbia.edu>
Date: Fri Dec 18 16:46:21 2015 -0500

   moar snippets
```

```

commit 63eb56b60685e849c785e4803502a41e6466d883
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Fri Dec 18 16:30:35 2015 -0500

   fancy graph decl tests

commit c8677a40abc7a7fe512283b7cd0947ea161247cd
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Fri Dec 18 15:11:31 2015 -0500

   some list issues let’s see if this works

commit c05f35dbd12858f34fd6c5651fcdc29737459833
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Fri Dec 18 15:09:37 2015 -0500

   forgot to look for the variable in symbol table

commit 62e66d575d7fac990b620d61ca09a8aff12771fe
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Fri Dec 18 15:07:47 2015 -0500

   trying to translate fancy graph declaration with braces

commit cb6a6faefe58e54da94567f181e435b6932c02d9
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 14:23:55 2015 -0500

   pattern match graphdef

commit b29dc5b9690a215b1c419e7f01f3deab276dde48
Merge: 34204ac d0465a3
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 14:19:33 2015 -0500

   Merge branch ’compile’ of github.com:adamincera/dots into compile

commit 34204ac2f3234d0b7d6df45605801419d531adb5
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 14:19:18 2015 -0500

   changed member var ’in’ to ’ine’ and ’out’ to ’oute’ because in was being
      read as ’in’ token keyword

commit d0465a3478259112395bcaf0347d16638894ab8c
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 14:19:00 2015 -0500

   fixed list printing to list the node->data field

commit 25d7ab88fae08bc30a66a26e4d2182eb6813d717
Merge: 67ae323 236cc64
Author: rgordon <rcgordon@umass.edu>
```

```
Date: Fri Dec 18 14:03:44 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

   merge`

commit 67ae32326f3fa51935aa5b7b133121ab8be8491e
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 14:03:34 2015 -0500

   implemented list printing

commit 236cc644fd1481f20044d65ce60e82136e5dda06
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 13:57:36 2015 -0500

   Mango and Hosanna added graphdef to parser, ast, sast, and typeconverter

commit 5e058c9668d8749e6243fd3fd050451ae2cc87a2
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 18 12:26:38 2015 -0500

   new parser rule to get rid of the shift reduce conflicts

commit 625f85060a04a0349a23491337320c8e9df29c27
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 18 11:39:21 2015 -0500

   added (expr) rule in parser

commit 22e2643b022c60544037b3bf971c8f092dbf6b7a
Author: hosannajfull <miramonte23@gmail.com>
Date: Thu Dec 17 21:11:38 2015 -0500

   hoho and mangy fixed the Member Expr calls

commit d30201c3e9da6751dcc1262f5eb3fde9249132d6
Merge: fad31c4 24fbbd0
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Thu Dec 17 20:17:57 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit fad31c402c8cf7c80ca6ed403e52c6d021bbf6f2
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Thu Dec 17 20:17:37 2015 -0500

   started changing access, membervar, membercall to expr * expr

commit 24fbbd0202ed9842dcfef014c78af1e2f9d4ece4
Merge: 0f7027a ccf099a
Author: Adam Incera <aji2112@columbia.edu>
Date: Wed Dec 16 23:47:59 2015 -0500

```

```
    pulling

commit 0f7027ade07a9cf860c76ab1ba8bb566bd6a2ce8
Author: Adam Incera <aji2112@columbia.edu>
Date: Wed Dec 16 23:47:25 2015 -0500

    changed TABLE_SIZE from 1024 to 256

commit fc1977814060823577d3ab59adb9b68038104b33
Author: Adam Incera <aji2112@columbia.edu>
Date: Wed Dec 16 23:47:00 2015 -0500

    fixed graph hashing

commit ccf099ab93d611c311cdefb5df559c256f92d740
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 15:35:58 2015 -0500

    tried to fix it so node_init function is called not in the global scope

commit ebec4ee3168d3c2ca0d15f31faf3742c25a82d4d
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 05:07:18 2015 -0500

    Well, sometimes you accidentally delete everything and then have to fix
    it.

commit 20654aef7e511e390636ce961a6d0df5b0df9fe8
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Wed Dec 16 04:48:10 2015 -0500

    debugged testing script, all tests should run but there might be some
       issues....

commit ee2701fb0addc027b61e1203395d62c919b0a832
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 04:35:44 2015 -0500

    moved tests around into folders

commit e2884a90f7038565f26ec541e46ebd41c6df9f38
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 04:20:47 2015 -0500

    modified testing script to allow for different folders of tests

commit c3b3e03c674555e0f497fe1d09cd2f6ddc1a7747
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Wed Dec 16 03:50:38 2015 -0500

    debugged analyzer.ml for binop... added a bunch of cases to suppress
       warnings even though in theory they should never be reached

commit 63a20ce5e217b712c968ff617c6433429726f83b
```

```
Merge: d5534df 080a55e
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 03:39:03 2015 -0500

    Merge branch ’compile’ of https://github.com/adamincera/dots into compile

commit d5534dfd1c2a630136012aa36cfc4d21c1652e79
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 03:38:29 2015 -0500

    rest of binops completed as much as I can right now

commit 15010a4ad9dbd60c3794f57f0bcd7dbde31be121
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 03:14:02 2015 -0500

    just realized a ton of binop rules could be condensed....

commit e4a66524e64c4bccbc057f790036ac35584c1c6d
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 03:09:34 2015 -0500

    condensed an Ast to Sast rule

commit e71901f3e0a7d4f561f99a5229e550849be6274d
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 03:08:16 2015 -0500

    in analyzer, add + subtract of binop first draft

commit 080a55e28293764b3cbe9baee90a953664164bf1
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Wed Dec 16 02:29:55 2015 -0500

    added gitignore for .o and .a files in clib from setting up

commit 68ba704aeabceda7eaeebaef1d8ae65ac1139daf
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Wed Dec 16 02:16:50 2015 -0500

    node decl fixes compile, let’s see if it passes tests...

commit 34dd071a40d688f88183cc72ece80b606863a350
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 02:11:45 2015 -0500

    first attempt at fixing nodes to not call init in global scope

commit a3326346b15821010d64e6e091f9bcd4df4c1614
Merge: 827726a e86213f
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Dec 16 01:45:11 2015 -0500

    fixing merge conflict
```

```

commit 827726a942872556f34340274bfe1c7c8755885c
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 15 23:15:49 2015 -0500

   mango and I semi-finished separating globals from regular statements and
       finagling the Sast tree into something more suited to C ast

commit 44a07ec1d6b0847f4fb5b6563fd36f2e29a13ec6
Merge: 128de36 cebc5a4
Author: hosannajfull <miramonte23@gmail.com>
Date: Tue Dec 15 19:34:16 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile

commit 128de3614a7e1d3c8050905394cbe8b87cb96c26
Merge: 189717c fe792c1
Author: hosannajfull <miramonte23@gmail.com>
Date: Tue Dec 15 19:33:59 2015 -0500

   merge conflicts handled

commit cebc5a456db42ec0fa0a239cb7ebb3c1d0c5b04f
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 15 19:25:49 2015 -0500

   removed extra inclusion of graph.h

commit 189717c584cc1aeabc3c89e9e6a6e27b4d090351
Author: hosannajfull <miramonte23@gmail.com>
Date: Tue Dec 15 17:15:42 2015 -0500

   program intermediate object update

commit fe792c17922d2232367c21f235c4beb328ac2039
Merge: e548ab6 3ccc4cb
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 15 16:24:40 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

   merge

commit e548ab68675c528d1e8764a1af5ff21cc67d6a08
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 15 16:24:30 2015 -0500

   changed Cast to only have stmts (no longer has exprs). moved the handling
       of print back into the translate_expr function in analyzer.ml -- this
       works because translate_expr now outputs things of type cstmt

commit 3ccc4cb7bae92a07041c703091feccc9545c08ac
Merge: 2179f4f a70354b
Author: Yumeng Liao <y.liao.2908@gmail.com>
```

```
Date: Tue Dec 15 15:59:21 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit 2179f4fae180032b9f2bd31477526da0c5680df4
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Tue Dec 15 15:58:58 2015 -0500

   tests for member functions and variables

commit a70354b6048bf771e10070790ed868deedde6bd4
Merge: 1e0d332 badb885
Author: Adam Incera <aji2112@columbia.edu>
Date: Tue Dec 15 15:42:25 2015 -0500

   pulling so i can push

commit 1e0d3329b41ab1dd68a8c364c0866c3316dc81c1
Author: Adam Incera <aji2112@columbia.edu>
Date: Tue Dec 15 15:40:46 2015 -0500

   UNTESTED clib stuff. graph copy constructor/comparison have been tested and
        work. everything else compiles but that's it

commit badb8852ff83ee0bf3c3fb8d4cb6e936db430b40
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 15 00:53:31 2015 -0500

   added more verbose error message for parsing error. made fixes to dijkstras
        test case. added stub rule for fancy graph declaration notation

commit c9bacf8bcdc014e77566fd5aad1afe8db40e44d8
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 15 00:30:26 2015 -0500

   Altered assign to take 2 cexprs which led to changes in how Assign is being
        called in analyzer.ml. also added 'N-' to the beginning of node
       printouts. fixed test files

commit 1cdd25222998f75ae20e400d51f5847f907ff594
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 14 23:47:21 2015 -0500

   removed the *.out line in the root's .gigignore file. DO NOT PUT *.out IN .
       gitignore --- WE WANT TO TRACK .OUT FILES IN dtest. added all the
       missing .out files

commit 93df1a2968a928d343cdf8eb86645b733b4835ae
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 14 23:44:45 2015 -0500

   -- Altered automatic variables to behave like normal variables, including
      adding them to the symbol table so they can be referenced later
   -- Restricted float printing to 3 decimal places
```

```
    -- Added function to convert Sast datatype to string.
    -- Added declaration checking to Ast.Id -> Sast.Id
    -- Implemented type checking and variable checking for Ast.For -> Sast.For

commit 363d2be5be1deb8308075497ea80049aaf8c2a93
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 13 17:39:17 2015 -0500

    enqueue dequeue remove min max implemented

commit 6c9ec9750e9d7d93fc5a6b4f2483feae6e3fe934
Merge: 4efcd4a 148bd83
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 13 15:58:40 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit 4efcd4aeea9a0be999c896a743c5647c7e58dd8b
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 13 15:58:35 2015 -0500

    added handling for checking types of print call

commit db899a5a079ce327c5a69b7ee1a5834dc4a634d0
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 13 15:55:31 2015 -0500

    updated readme with install instructs

commit 148bd83ba1965c9d41199b5c7ecc3ba1bd9e8cf9
Merge: 45deb11 9d92232
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 13 15:48:18 2015 -0500

    Merge branch 'compile' of github.com:adamincera/dots into compile

commit 45deb119ca6df2f359b898e8522655ce906e487b
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 13 15:48:01 2015 -0500

    all warnings are done

commit 9d9223243a4b6527e2f7cd9264ca2a8d570f95da
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 13 15:47:21 2015 -0500

    snippets for printing list, and for adding strings

commit 6e4007b68e9fbeae4c1a0490b95d0ed8a77a0bba
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Dec 13 15:02:08 2015 -0500

```

```
    snippet for printing dicts

commit 414869db83a3ecd47883b3ed0b241f706422c5c7
Merge: 058b610 0c9a2d5
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Dec 13 14:18:14 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit 058b6100fc9189bee8f857f3772bf734372d3f94
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Dec 13 14:17:51 2015 -0500

    forgot to add dijstras test

commit 0c9a2d5bc22f567075fe1ceb53a52ade633aee8d
Merge: 412f6f8 80d5136
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 13 14:11:19 2015 -0500

    merge

commit 412f6f857d6bf3f98ac6c4805059ec51819a4d93
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 13 14:10:26 2015 -0500

    new test case and splitting up types different in get_fmt_val funct

commit 80d51367159b8e619a4febf3f9078266e5ebf55c
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 12 23:40:56 2015 -0500

    tests to call range function in for loop

commit 0d7048f5e04d1fe7cae8623cf3b2dcbc045c9d3e
Merge: 718adb1 80b44ce
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 19:34:26 2015 -0500

    pulling again again

commit 718adb1b52436c2b14dc809bd9256ce3249991d3
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 19:34:03 2015 -0500

    implemented for in dict c translation

commit 80b44cebf9fbb4b476d82a4ac0419eaa98c62a28
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Dec 12 18:56:14 2015 -0500

    tests for calling range function with 1 or 2 args

commit 65e260796a356dde93d242e8947c358567733ddb
```

```
Merge: 6ee720b 8640189
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 12 19:02:40 2015 -0500

   call works changed fdecl

commit 6ee720b1987cb993554ebb4d4550e5469d12ad62
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 12 19:01:25 2015 -0500

   call working with functions ya bish

commit 8640189a12d55691cd003153cc0727b906a45670
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 18:35:42 2015 -0500

   translated for key in dict

commit 404f898340b17f73e3f789ddd0daeca5771db5d1
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 12 16:58:26 2015 -0500

   fixed printing of nodes. adding setup rule to Makefile that will run 'make
       library' inside clib. altered print statement in runtest.py

commit cedb611e9038ef21304b76736bfa0eb229796837
Merge: edf0a2b 5460531
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 14:49:23 2015 -0500

   pulling again again

commit edf0a2b1cbf67312aae7806b21b649aaf4e66511
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 14:48:58 2015 -0500

   tested other dicts

commit 5460531569dddf342c1d3f488725a3a52462873b
Merge: 00aeabe d309f87
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 12 14:46:25 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile

commit 00aeabe20d67e08e2faa6b2c947256692f5fdfc4
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 12 14:46:08 2015 -0500

   fdecl working thanks Ratchel:

commit d309f873f74597366614069cf1ede69018feaaf2
Merge: edbe506 1659bbe
Author: Adam Incera <aji2112@columbia.edu>
```

```
Date: Sat Dec 12 14:39:51 2015 -0500

   pulling again

commit edbe506355854ec845405f5fd819a3e06403f70d
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 14:39:23 2015 -0500

   fixed num putting

commit 1659bbe1f355600c1649536861ae0f11c9b9e022
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Dec 12 14:36:25 2015 -0500

   Fixed wrong folder name, now works as intended

commit a9995dcac8e4472fc296037d668118885834a059
Merge: 0013e30 aa3e2f6
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Dec 12 14:31:28 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit 0013e302c3ac80301d65bad6575fefedba8f61bd
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Dec 12 14:30:50 2015 -0500

   applied test script changes to the right branch this time....

commit aa3e2f6d778ef950a6fd8fba8fbafcc9fa1b9be5
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 14:24:53 2015 -0500

   fixed snippets typo

commit e86213f47aa084a3af79284131f2df6159400ae4
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 12 14:12:34 2015 -0500

   added handling of negative tests to testing script, deleted old one

commit 4337dd1276c9ed89ea3f26233989e4a61e119a79
Merge: d2580fe eaa8d76
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 13:52:06 2015 -0500

   pulling

commit d2580fe4e745049ffa2f72bcc0957734802ebf7f
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Dec 12 13:51:14 2015 -0500

   added snippets for dicts

```

```
commit eaa8d761e5fbba8a8215b9bde8385be1ce7f9a5f
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 12 13:40:35 2015 -0500

   node def

commit ab98dd2657da84ae08f08a81de58b8409cc79180
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 11 23:39:38 2015 -0500

   fixed parser rules

commit 0132432d45b41b2f3638e70ee63ae34b5e615558
Merge: 6495cd7 f9212da
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 11 23:24:55 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile

commit 6495cd712f46e616f18586c4ccd3e3cf7e314408
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 11 23:24:52 2015 -0500

   merger

commit f9212da61a12ab722e11e15434916b74ca1c1df1
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 11 23:24:03 2015 -0500

   mango added initial work for new type NodeDef

commit 16fcf935f07906a69977b41b20728c7218e73bd8
Merge: 3d6bcd0 224155e
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 11 23:14:35 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile

commit 3d6bcd07de58e260113b08852ca0b295351617da
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 11 23:02:49 2015 -0500

   finished typeconvertergit statusgit statusgit statusgit statusgit status!
      maybe

commit 224155e98e695103f09bbbc935871c63ca69783d
Merge: 060a7ea 2c546ac
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 11 21:09:13 2015 -0500

   Merge branch 'compile' of https://github.com/adamincera/dots into compile

   merge

```

```
commit 060a7ead3c5fad95bfc0a5c13896b67b905abc49
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 11 21:09:04 2015 -0500

   altered the way print translation works

commit 2c546ac6f00ed8753fb6f4b7d6ee65c3d9458abe
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 11 14:00:49 2015 -0500

   s_expr done

commit 702d94d4e6c9ce2c2a2ecfea18d6a5d33534e902
Merge: bc0e6df 804f95a
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 11 11:03:05 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile

commit 804f95a8fd7049d432d83c9622bd29ad6c6c3fb5
Author: Adam Incera <aji2112@columbia.edu>
Date: Tue Dec 8 03:58:15 2015 -0500

   dict is working for string keys. more testing needed for floats and void *

commit bc0e6df34c3790fbb356f070d4158a8b24109a47
Merge: be0075e 56c1b47
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 23:56:28 2015 -0500

   Merge branch 'compile' of github.com:adamincera/dots into compile

commit be0075e4cae5d69915ab1f29fc41ec2ef63a2300
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 23:56:10 2015 -0500

   Ast.MemberVar complete

commit 56c1b47f0ff3caf276976ef7cbb381ac998698a5
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 7 23:04:29 2015 -0500

   removed commented out code from gdc

commit c1542b69ae6eadc5b43b17cbece31ea7868c1d12
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 7 22:54:32 2015 -0500

   added library and objects target to makefile

commit b45353a3ec488af3e3e9c45c60283091147e65a0
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 7 22:46:07 2015 -0500

```

```
    adam fixed the c linking for gcc

commit c6342f0b2a23def1678e63351ab293ba2debf3cb
Merge: cff0152 5b9ba2b
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 7 21:55:21 2015 -0500

    pulling

commit cff0152ceb09be447200da744efe2ff661ad8d19
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 7 21:54:48 2015 -0500

    changed node data type to char *

commit 5b9ba2beaa776a7542123a479003cb6b3e095d91
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 7 21:37:15 2015 -0500

    fixed type used in Id to C translation. fixed include headers for clib
        files. removed extra line from gdc. added some new test cases

commit 56816a18501f8e88ee8102d533fe6030b231e256
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 21:09:29 2015 -0500

    parser changes to DictLiteral

commit 17b558d35693777638a4c3dd3272424c10c6e33f
Merge: 5abd029 0930670
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 21:06:42 2015 -0500

    Merge branch 'compile' of github.com:adamincera/dots into compile

    merge

commit 5abd0294bedb10de374cbc1a2a5b9f849e7ecb61
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 21:06:25 2015 -0500

    made functions part of statements and fixed compilation issues

commit 093067066d1d9b27985a3f8d45ea5a3988fcca30
Merge: 7854be5 9c5fafc
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 7 20:37:43 2015 -0500

    merge

commit 9c5fafc94348891b932070f64a05b2784cd5fb57
Merge: d538667 3534164
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 20:28:14 2015 -0500
```

```

    Merge branch 'compile' of github.com:adamincera/dots into compile

commit d538667eb8eedd8190cf797a36a3ea96b88c40c2
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 20:27:55 2015 -0500

    rachel fixed dict literal kvl

commit 7854be57e956bd97b624ebe57a33dbf915164bfe
Merge: e23e4ff 3534164
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 7 20:18:39 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit e23e4ff78cdac6c9d0c4362cdd9bc85f10f29f6a
Author: rgordon <rcgordon@umass.edu>
Date: Mon Dec 7 20:18:24 2015 -0500

    added new clib include libraries. made for n in node expr translation
        simpler

commit 353416473c41baca89e78d95469e76fffbe20d46
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Dec 7 20:16:47 2015 -0500

    renamed translations.c/h to something more useful, list.c/h

commit a5b5365831a1e4c372890a4fc84d2940f21f2c3b
Merge: 76ef60f 7f12856
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 19:25:35 2015 -0500

    you used to you used to

commit 76ef60f591f9c92117be7436c8b843d0ec7ce6f3
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Dec 7 19:17:55 2015 -0500

    you used to you used to

commit 7f12856fa664c8237d18f982f77948a94aa3d33f
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 20:31:02 2015 -0500

    added ListLiterals as a type to Ast and Sast. moved check_list handling to
        occur at the ListLiteral level. removed (commented out) AssignList type
         of Ast and Sast and moved all handling into just simple Assign type. (
        possible b/c of the new ListLiteral type)

commit a7ded13546fc3ec12ff8323d652f4238a55d9b75
```

```
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 19:55:28 2015 -0500

    implemented block to C. other small implementations

commit 474550cdadd1259037320beae65cf1018abd02b6
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 19:33:46 2015 -0500

    fixed Assign def in parser (didn't end in a SEMI). moved the addition of 'v
        ' and 'f' for func and var names to be added in analyzer instead of
        translate (so that you can call specific function names / var names as
        well). implemented node declaration.

commit a6474d213ab3b621ec7cea7425eae043d3571367
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 19:07:09 2015 -0500

    implemented rest of For loop to C except for 'dict' data type. implemented
        while loop to C. added some simple test cases (still need expected
        output files)

commit 83570804fc36cef76890997df095a4418f2bf203
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 18:36:36 2015 -0500

    added clib/graph.h to the list of includes

commit 6c26461ac3e4e6bd2f381e18763c350c5a270624
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 01:57:18 2015 -0500

    begin fleshing out for loops to c

commit ec8d0fe6b2d00c844fd4e8823999814fb2123e2f
Merge: 7e2c025 28e5c4f
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 01:09:14 2015 -0500

    pushed to wrong branch. fixing

commit 7e2c0256afce14f33fd11e26c48c9e696831b429
Merge: f51e890 caa274e
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 01:07:18 2015 -0500

    merge conflicts

commit f51e89049dde73ead4ea9719c6c4c36c1edb6e45
Author: rgordon <rcgordon@umass.edu>
Date: Sun Dec 6 01:05:27 2015 -0500

    parser rule for ListDecl was wrong

```

```
commit caa274e0773697aae8dfb4c5652ba7224f5d989b
Merge: a88b704 8ad664f
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 6 00:50:37 2015 -0500

   fixed the merge conflict

commit a88b7046b47f4cbf6357224307199ace0c553548
Author: hosannajfull <miramonte23@gmail.com>
Date: Sun Dec 6 00:30:32 2015 -0500

   ratchel

commit 8ad664fc79e6d32c9d9f45322742913721f89c39
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 5 23:44:11 2015 -0500

   initial work on getting list to c translation

commit d9636bba3a702decf5f94d8c181f20f497f785bb
Merge: f4ffe22 4166f5e
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 5 22:16:49 2015 -0500

   Merge branch ’compile’ of github.com:adamincera/dots into compile

commit f4ffe22de05f00ce2b2a49d5dd66f6eca4e6b2c4
Author: hosannajfull <miramonte23@gmail.com>
Date: Sat Dec 5 22:16:41 2015 -0500

   assignList

commit 4166f5ea1f51a4acdf590e18f8ccfc8b7c99b48c
Author: rgordon <rcgordon@umass.edu>
Date: Sat Dec 5 21:27:00 2015 -0500

   fixed various bugs in typeConverter

commit 0aade866d412c9c180319ec808097a2bb688f38f
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 5 02:34:24 2015 -0500

   added == semantic checking, fixed more spacing issues

commit 0662952396a30e386eaf46e23667407fc57c53b3
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 5 02:20:57 2015 -0500

   added some logical operations semantic checking (last commit meant to say
      checking, it’s late), fixed spacing

commit 6d177409debabd6f4baa9096cc892ff156b8a826
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Dec 5 02:10:07 2015 -0500
```

```

    added subtraction semantic parsing, made error messages for semantic
       parsing more helpful

commit 3eb962f34557f29865c099541a26b59bfa73a08a
Merge: 9ecf51c bb6f307
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 4 22:04:38 2015 -0500

    fixed merge conflicts. but it doesn’t compile so we need to fix that

commit 9ecf51c03c742b205dcd278a37cd960f0473e637
Author: rgordon <rcgordon@umass.edu>
Date: Fri Dec 4 22:00:53 2015 -0500

    fleshed out the sast to cast translation. fixed the compile file and fixed
       the Makefile to actually use it

commit bb6f30715f1b2d30cdf551763a1d9629f781ff6f
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 4 22:00:45 2015 -0500

    pattern matching for type checking

commit 9cac4d89a0623e6b819a9ea8db4777e52b75b517
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 4 19:27:05 2015 -0500

    Mango placeholder types in analyzer to get it to compile

commit 974efff87bff31cc6037411fe34ba91f35bad54a
Merge: b75b609 369140f
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 4 17:43:29 2015 -0500

    Merge branch ’compile’ of github.com:adamincera/dots into compile

commit b75b6091351fa126f8517ef91bc5aadda31dbc63
Author: hosannajfull <miramonte23@gmail.com>
Date: Fri Dec 4 17:43:16 2015 -0500

    comments on what needs to change

commit 369140fecf094e261c270a5cf7956db7c20887ed
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Thu Dec 3 01:15:02 2015 -0500

    Started fixing translate_stmt in analyzer.ml

commit 1e77665dd8e4dcb297dccde534141321596c1736
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Thu Dec 3 00:04:52 2015 -0500

```

```
    fixed translate.ml to take c statement lists rather than strings, next up
       is analyzer

commit a7f90380bbf338a7412d402f00fc7eddd147679d
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Wed Dec 2 22:49:59 2015 -0500

    added bit to translate whole program calling predefined functions, cleared
       up library stuf

commit 4304b02a6fe0ab5cbd2672a091356ab58bfe36be
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Wed Dec 2 22:24:15 2015 -0500

    added library translating function, still have to clarify it though

commit 852dbf204844b55e0f9f6e72a6dcf7eddc7bcca5
Merge: e038853 eda78c0
Author: rgordon <rcgordon@umass.edu>
Date: Wed Dec 2 21:55:58 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit e03885327826c00a0e4f8829375f8140150ff544
Author: rgordon <rcgordon@umass.edu>
Date: Wed Dec 2 21:55:47 2015 -0500

    initial work on creating the C syntax tree

commit eda78c06a1fb165251d6bd39fc9513ad6ccb6105
Merge: aaeabf8 1517189
Author: Adam Incera <aji2112@columbia.edu>
Date: Wed Dec 2 21:24:54 2015 -0500

    merging

commit aaeabf831de770eb3541ffc0c29a16edc73f3671
Author: Adam Incera <aji2112@columbia.edu>
Date: Wed Dec 2 21:23:28 2015 -0500

    added list things

commit 151718960842f5c5fd54504c0a8dfa707a326296
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 1 17:47:06 2015 -0500

    removed separate rules for LogAnd and LogOr and put them into regular
       Binops (added an operator for each of them)

commit 9f80b6a3080dec6ae2291b33f92353856715c37e
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 1 17:21:29 2015 -0500
```

```

    fixed expected output for string literal and num literal test cases (should
         have been nothing since there is no print statement). added output
        file extensions to gitignore

commit 4b8473972e9710f220803e5b7961b02683c8d56f
Author: rgordon <rcgordon@umass.edu>
Date: Tue Dec 1 17:11:54 2015 -0500

    the .gitignore incorrectly had 'gdc' inside. switched it to 'dotc'

commit 5d0b6c889a2bf1a3430d659794525146a88cdf9b
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Tue Dec 1 16:27:45 2015 -0500

    testing script now prints nice summary of passing tests in alphabetical
        order

commit 2e62d11ec597c9e4cb31c54368288482ff510939
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Tue Dec 1 16:12:14 2015 -0500

    added .out files for test script from tests written yesterday on simple
        list + dict decls

commit 3da7cf18591da8b4df58010773b2f0454cd578a6
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Tue Dec 1 16:09:38 2015 -0500

    made requirements.txt for testing script, updated test guide

commit 81293310c62073cac8e1bdbeb545f85c7ebb5c88
Merge: 99ac50b 42324b2
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Tue Dec 1 16:00:59 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

commit 99ac50b4a1de5ada95346b02715b6a109241a83a
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Tue Dec 1 15:54:32 2015 -0500

    added test for list library functions

commit 42324b22066955980dcc5a3e1a38fb7e47695481
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 30 21:59:53 2015 -0500

    Mango removed dict decl and made dict declaration vdecl

commit 8e2961054a9c38dbd2f2c2cc7734d61fb67f6e40
Merge: b09db0b dc45603
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 30 21:22:54 2015 -0500
```

```

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit b09db0bb0ff8ad88d7b3d101a238e13713472aee
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 30 21:22:29 2015 -0500

    added code for translating a list declaration. BUT it still needs the C
        snippet. also changed permissions of runtest.py and testall.py to be
        executable. added new simple list test case. TODO: add expected output

commit dc45603dc4d9bc5f81fa184b4c7d150f3097f834
Merge: 3845a2c 0b4375a
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Nov 30 20:08:34 2015 -0500

    rachel pushed stuff :(

commit 3845a2c583314788279fb8d3f0d888c9494e3113
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Nov 30 20:07:53 2015 -0500

    added snippet for copy

commit 0b4375aa0a87d3171f472e525263268e40dad2ba
Merge: 15ad512 a6542ab
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 30 20:00:04 2015 -0500

    Merge branch 'compile' of https://github.com/adamincera/dots into compile

    merge

commit 15ad51228551265137b92e09d8e02f8b8ce9c345
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 30 19:59:57 2015 -0500

    added comments. changed DictAssign to contain 2 exprs

commit a6542ab60c7924ca40c35ea96f64a57da7d98535
Merge: fdf472e eba852f
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Nov 30 18:11:15 2015 -0500

    pulling clib into compile

commit fdf472e316bc4ba01d0ad1749488c52e788cedbd
Merge: ed5e1bb 701c540
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Nov 30 18:10:40 2015 -0500

    pulling compile
```

```

commit 701c5404a5faad318a20b83bc3a3d40962f7460f
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 30 12:22:43 2015 -0500

   finished / fixed Vdecl handling in the new format

commit 28e5c4f81a6bfc1996ccb7da8a33121f5509277a
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 30 11:38:50 2015 -0500

   changed final report Makefile to refer to the correct file

commit eba852f71c2a21b682eeef38c9f42c5da4e33207
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Nov 30 05:59:36 2015 -0500

   added copy function, updated snippets

commit a01fa3fb3978348009678cf9c15a59f69a926493
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Nov 30 05:20:26 2015 -0500

   fixed plus and plus_equals functions

commit c2c57edc6a152d5151369a746fda7a323dd0a7a9
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sat Nov 28 23:50:12 2015 -0500

   Made folder final-report with rough draft .tex file with basic sections for
        our final report

commit 91f6fc3860d7463cc424bd94924395ed7b3412b3
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 27 23:56:58 2015 -0500

   removed last reference to old hard coded variable tables. removed
       extraneous comment

commit b9c22a26e980fb8226119590bfbd542630a47d6c
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 27 23:47:46 2015 -0500

   variable declaration now works with the new environment setup. the print
       function still doesn’t work. unclear whether the scope variables are
       truly updated

commit e263632981d5e3c38dba1a619ff6c7dd1e113fef
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 27 23:40:25 2015 -0500

   altered analyzer.ml to add functions that convert the Ast to the Sast.
       Fixed incorrect types in Sast. Created a translation environment
       variable that can be used in semantic analysis.
```

```

commit 02ef7cc396bd9e1f2606f5a6299b2c8206a148ab
Merge: ad45044 92df7d0
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 27 10:54:24 2015 -0500

   resolved merge conflict

commit 92df7d0d9a148d74f1bef3db2b27cc63cbf18807
Author: hosannajfull <miramonte23@gmail.com>
Date: Tue Nov 24 12:52:12 2015 -0500

   env var integration:

commit ad45044715a11abe389ebf0a6cdceff619a924f7
Merge: 977cabb d495509
Author: rgordon <rcgordon@umass.edu>
Date: Tue Nov 24 11:18:20 2015 -0500

   merged hosanna's changes

commit 977cabb8367842f870a34b6b69572e3674d9b46b
Author: rgordon <rcgordon@umass.edu>
Date: Tue Nov 24 11:14:43 2015 -0500

   additional work on creating env var. new test cases

commit d49550956b2f133f35e1e96edcfd7424d6699d77
Author: hosannajfull <miramonte23@gmail.com>
Date: Tue Nov 24 10:30:55 2015 -0500

   files merged

commit cf1e0bee61b1886bef74c2fb9c455d90305e7b67
Merge: 1dc5b83 82761b5
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Nov 23 20:25:20 2015 -0500

   merged all files

commit 1dc5b8330463c839816e729b2f348cbe8fdfc262
Author: hosannajfull <miramonte23@gmail.com>
Date: Mon Nov 23 20:11:25 2015 -0500

   sast

commit ed5e1bb5198f1657db9a6e939b25408e14c7dc17
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Nov 22 17:43:22 2015 -0500

   wrote some snippets for translation

commit 82761b54a30454e7502440490f92a0a034a37e24
Author: Yumeng Liao <yl2908@columbia.edu>
```

```
Date: Sat Nov 21 23:54:06 2015 -0500

   added some dict tests

commit 380ff23d6dfd8c7034fe5dc72d635fe79a245486
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Nov 21 21:38:48 2015 -0500

   list declaration, assignment, access tests

commit 3681084fcf74ef5086f559d476b917141f7072be
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 16 10:19:56 2015 -0500

   changed num literal parsing in scanner to include floats. removed
      unnecessary comment from analayzer.ml. altered num-assign test to
      assign a negative float to the variable. renamed testdots.py -> runtest
      .py. renamed tests.py -> testall.py.

commit d64c91c1d6cf8a245d1544a93a23f3890b74c881
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 16 01:06:14 2015 -0500

   fixed parsing rule for assignment operator. added test case for assigning a
       num (which passes - yay). fleshed out translation of Assign expr

commit 1e255c94d891c1f74a67c8433ab6db3fb462c72f
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 16 00:46:45 2015 -0500

   added placeholders for each case for translate_expr and translate_stmt to
      get rid of all those annoying warnings

commit b280b35987fc3804ee2343f08e1dfeb7fa2d658b
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 16 00:30:57 2015 -0500

   cleaned up print parsing a little by adding a function that figures out
      what format string type each expression is

commit 1fdfb02362e6fffaf41709076176c49df8efea04
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 16 00:17:02 2015 -0500

   altered .gitignore to only ignore those particular file extensions for
      filesi in the documentation folder. before it was ignoring .out files
      in dtest, which we DO want to be able to commit to the repo

commit ae6f91880717bb83bafde40b418456c196af73be
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 16 00:13:06 2015 -0500

   added check to testdots.py to make sure the .out file exists for a
      particular test case. altered way function call translation works:
```

```
      moved logic from translate.ml into analyzer.ml in the translate_expr
      function; changed it to printf with a format string so that things that
       aren't of type stirng can be printed

commit 8bb993ec94c2e35fe89f29e8066cc18e25312c26
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 23:03:18 2015 -0500

   altered variable declaration so that it maps local variables to indices and
       declares l1, l2, .... ln instead of using the original variable name

commit 785c00c266bba9928af7d67f3ad49af2cb620a2b
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 22:42:37 2015 -0500

   fixed adding variables to locals_indexes (involved turned locals_indexes
      into a ref). added placeholders for unhandled cases in ast.ml

commit 5b95741260579dfa9ebed11534fcad55ab69c376
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 22:03:56 2015 -0500

   changed it so that executables are compiled to files ending in '.exec' so
      that then the clean flag can properly remove those files. also changed
      it so that executables are compiled into the test directory instead of
      straight into src

commit 3b3b362812a06df371599e7e24be4fd2daff1e26
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 21:56:04 2015 -0500

   added rule to Makefile to delete *.outgdc files. changed argument parsing
      in testdots.py to use argparse module. fixed testdots.py so that it
      doesn't run the non-existent exeutable if compilation failed. fixed
      expected output for multi-hello-world.dots

commit 704d7c3f3533b7c6cde378c877e1e9c60aac8fe1
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 21:30:24 2015 -0500

   added rule to clean that removes all .c files in dtest

commit c956fdd02fdc3667565233bba8af514674a4080d
Merge: 84337d9 b40061d
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 21:29:10 2015 -0500

   integrated parser branch into compile branch

commit b40061d38da265f9a6fae510fde6948ac43d05ee
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 18:36:59 2015 -0500

   got rid of random executables
```

```

commit a5421a1c36064766427f6d458f94cf26120f1220
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 18:32:59 2015 -0500

   added -B flag to diff command that fixes tests. changed testdots.py so that
        the flag -c will remove files; otherwise all output files are kept at
       the end of the test

commit c3f960bccaccb99b933a56f9b89929dcd6286928
Author: Hosanna <miramonte23@gmail.com>
Date: Sun Nov 15 18:23:53 2015 -0500

   wording change

commit f4b3ed3c7890de962e8b8ee630a6269fbb0aada1
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Nov 15 18:21:32 2015 -0500

   added range function in translations.h

commit d99bc7ff95f490db2b12fb97ac5f88aaca2ee788
Author: Hosanna <miramonte23@gmail.com>
Date: Sun Nov 15 18:08:38 2015 -0500

   set -e added

commit 29f9e165d2b8074a6d5955a52c85bce12365345d
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sun Nov 15 17:44:09 2015 -0500

   fixed testing script diff checking

commit 84337d9cdd9545e043b480f08442b2354b46f60c
Author: Hosanna <miramonte23@gmail.com>
Date: Sun Nov 15 17:41:04 2015 -0500

   progress on vdecl

commit ff31564d51e4f26eab7d43f9a33df0023a161ac2
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sun Nov 15 17:32:01 2015 -0500

   test harness works but output file is being poopy

commit 0441d1ddb364bee8251369400784309e319c4af9
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sun Nov 15 16:48:35 2015 -0500

   changed the makefile to NOT accidentally delete all of our c libraries
       .......

commit 071e1ca4c36f2c73f2b8e385202d32ebbc1d28b6
Merge: 3937078 b2044a2
```

```
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 16:47:06 2015 -0500

   Merge branch 'parser' of https://github.com/adamincera/dots into parser

   pull merge

commit b2044a278c7aa17c49b92a375ef0d2ba746d429a
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sun Nov 15 16:31:00 2015 -0500

   accidentally deleted our shell script...

commit 2cee4ced9cce0b744298aecfd99594c994c88f15
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sun Nov 15 16:13:23 2015 -0500

   make clean removes executables too

commit c3daddf6a5e209aa7fea1e743b212f13d97bca2a
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sun Nov 15 16:11:34 2015 -0500

   made .gitignore more comprehensive

commit fc1c7db32cdc94cfd8a45e23d23fd0aa6a80da53
Author: Hosanna <miramonte23@gmail.com>
Date: Sun Nov 15 15:42:10 2015 -0500

   working built in funky funcs

commit 3937078f2fafdd69590015df521c3366d9cc4d00
Merge: ccb3212 f500303
Author: rgordon <rcgordon@umass.edu>
Date: Sun Nov 15 15:18:43 2015 -0500

   Merge branch 'clib' into parser

commit ccb3212c429b77717e9f176324f2bc003b20edf5
Author: Hosanna <miramonte23@gmail.com>
Date: Sun Nov 15 15:17:44 2015 -0500

   deleted interpret microC copy

commit 5a9dc78ab564eea5110b41969a5560b18e0b8dc2
Author: rgordon <rcgordon@umass.edu>
Date: Sat Nov 14 16:41:05 2015 -0500

   fixes to shell script and hello world test files

commit f5003033ddd4b56572f74e97343a332504ff7d3a
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Nov 14 16:39:43 2015 -0500

```

```
   restructured node removal, added edge removal

commit 8828505f14ec1db4cf282a1e039d9cf12837dab2
Merge: bc82f3b 461e632
Author: rgordon <rcgordon@umass.edu>
Date: Sat Nov 14 16:11:03 2015 -0500

   Merge branch 'parser' of https://github.com/adamincera/dots into parser

   aaaaahhhhh merges

commit bc82f3bfd0f39a2183f69d7ed71ee7d6012dfacb
Author: rgordon <rcgordon@umass.edu>
Date: Sat Nov 14 16:10:47 2015 -0500

   OMMMG HELLO WORLD WORKS

commit 461e632be28d37408b8f14560d210bd284b2106f
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Nov 14 15:11:01 2015 -0500

   test automation script and some tests and out files, option to print diff
       with -k flag

commit fbdd4595a863649ea36f01ab0939510e3987e8ba
Merge: 78b3b2a ca78e91
Author: rgordon <rcgordon@umass.edu>
Date: Sat Nov 14 14:58:19 2015 -0500

   Merge branch 'parser' of https://github.com/adamincera/dots into parser

   whoops didn't pull

commit 78b3b2a801723cf33e2014fdd8556e2a17915580
Author: rgordon <rcgordon@umass.edu>
Date: Sat Nov 14 14:58:09 2015 -0500

   fixed function declaration parsing and pretty printing

commit ca78e9192ff7440561213a32031c0465561f2f1d
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Sat Nov 14 14:38:27 2015 -0500

   hello world endgame tests

commit 1589aaae506dc06ec6be7d4204cbc46e81ae116c
Author: rgordon <rcgordon@umass.edu>
Date: Sat Nov 14 14:05:26 2015 -0500

   fixed parser definition of an fdecl. started writing compiler

commit 9077169727517d3c4ac94fe63e52cbc911b7fa65
Author: rgordon <rcgordon@umass.edu>
Date: Sat Nov 14 12:13:10 2015 -0500
```

```

    added differentiation between string literals and num literals.

commit 7b5cabcb0d45e47a0195f693b233315a4096e399
Author: rgordon <rcgordon@umass.edu>
Date: Sat Nov 14 12:01:01 2015 -0500

    fixed string literal parsing

commit 6bed058244e6c677d7348cbfdb9387bcfb461e75
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Nov 14 03:49:29 2015 -0500

    fixed subtract! was actually just yet another bug with remove(). also
        deleted .swp file that snuck in the last commit.

commit 88045d86d7d288a5d123d1f15749a34011fe5918
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Nov 14 03:39:53 2015 -0500

    accidentally committed .o files and executable :(

commit debb0afcd7f243fa0db73a8b88d138098a123f58
Author: Adam Incera <aji2112@columbia.edu>
Date: Sat Nov 14 03:34:01 2015 -0500

    c library! all functions seem to be running without memory leaks except
        subtract

commit 3bc1b163c9ce6093e95e2b3d5ba74b58517416d0
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 13 18:14:07 2015 -0500

    removed vars field from program type

commit fded5410d7d3f3cbf97c309c46cc60cc8d296985
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 13 18:00:39 2015 -0500

    altered variable declaration so that it’s now just another kind of
        statement. added types to ast.ml to support this change. fixes to
        pretty printer

commit 12f44aed7f27792e86b30ebea79470c5e0911d74
Merge: 08b8e03 0c28e1b
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 13 14:52:29 2015 -0500

    Merge branch ’master’ into parser

    integrating updates to lang-ref-man from master branch into parser branch

commit 0c28e1bdfd2eb9bcfc74e051dd183465502cfbcd
Author: rgordon <rcgordon@umass.edu>
```

```
Date: Fri Nov 13 14:50:38 2015 -0500

   re-added the list and dict section to the lang ref manual. PLEASE don't
      overwrite/delete these changes again. --> proof of the need to double
      check git diff before committing

commit 08b8e03718fae7421ee8b01c7bbb0b3beba84e0f
Author: Hosanna <miramonte23@gmail.com>
Date: Thu Nov 12 23:42:28 2015 -0500

   pretty printer with types

commit 2a6e5d26ddc27ac9be159f3c68cb6a232ac3c0d3
Author: Hosanna <miramonte23@gmail.com>
Date: Thu Nov 12 12:52:49 2015 -0500

   commenting the parser so we know what we need tuples of

commit 20f46314c2d4fbda1401438678da20d4a406eae2
Author: rgordon <rcgordon@umass.edu>
Date: Wed Nov 11 11:54:38 2015 -0500

   fixed Ast.expr / string error. fixed reduce / reduce conflict. can now feed
       dots files to ./dotc

commit 6785a4f2cc625656664f9b805c9428835db4c79e
Author: Hosanna <miramonte23@gmail.com>
Date: Mon Nov 9 19:29:22 2015 -0500

   added up to date compilation efforts

commit ec029757f4f8f90110182fe4ec29fe82c063f583
Author: Hosanna <miramonte23@gmail.com>
Date: Mon Nov 9 18:02:35 2015 -0500

   compiler

commit b53b098c17fb51acdcbf4b5b7dc85f4487461d36
Merge: 921f9b0 cf233e0
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 6 11:54:49 2015 -0500

   Merge branch 'parser' of https://github.com/adamincera/dots into parser

   merge

commit 921f9b06078ebddd7847cc72db32f6d1745ff4a9
Author: rgordon <rcgordon@umass.edu>
Date: Fri Nov 6 11:54:36 2015 -0500

   added an analyzer.ml file to act as our compiler file. fixed some syntax
      errors in Ast.ml.

commit cf233e03b463ffad062fd243462b92c26bdcf470
```

```
Author: Hosanna <miramonte23@gmail.com>
Date: Wed Nov 4 14:36:04 2015 -0500

    added a baseline interpreter

commit bf58fd6c1e116697951f3bea33880f1daf0afdab
Merge: c7de03b f94390c
Author: Hosanna <miramonte23@gmail.com>
Date: Wed Nov 4 14:29:18 2015 -0500

    Merge branch 'parser' of https://github.com/adamincera/dots into parser

commit c7de03b76bdc1b0d029249c8f6db856386d6f56f
Author: Hosanna <miramonte23@gmail.com>
Date: Wed Nov 4 14:29:00 2015 -0500

    added microC version of our code

commit f94390c2063798a81039588751208dcf44bfed93
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Mon Nov 2 22:34:44 2015 -0500

    added method to write tests that should fail and print that it should have
        failed but passed

commit df650eeddb0244037cb4291fa1bed7edeb8406e7
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Mon Nov 2 21:57:54 2015 -0500

    worked out a few list and dict tests

commit 29021b3f4c78356aa75b0634681295904022e597
Merge: a2de0d7 aa14b69
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 2 21:50:46 2015 -0500

    Merge branch 'parser' of https://github.com/adamincera/dots into parser

    merge

commit a2de0d72728744b73412f9edbea98de147490761
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 2 21:50:32 2015 -0500

    changed list decl assignment to require a actuals_list instead of
        formals_list, meaning that expressions can be assigned to lists.

commit aa14b69aac7dae922f6596b5759476ef76ae8edb
Merge: 683a340 a6549cc
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Mon Nov 2 20:40:38 2015 -0500

    Merge branch 'parser' of https://github.com/adamincera/dots into parser

```

```
commit 683a340ab4030fdd155d22942f3315047dbe4f42
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Mon Nov 2 20:39:50 2015 -0500

   Tried to keep tests informative but ran into some issues with tokens, talk
      to Rachel

commit a6549cce9f339bd298a8871848ee952cd460b7c4
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 2 18:42:35 2015 -0500

   allowed key of assigned dict declaration to be a literal

commit 0ef877da379e979c1383a67295798485737b6384
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Mon Nov 2 17:36:24 2015 -0500

   improved testing script to use python subprocess and various useful flags,
      documented in test_guide.txt

commit 9bb86de36eaeb5489b2861fb649499bebc1133ae
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Mon Nov 2 15:01:05 2015 -0500

   Moved test scripts to upper directory

commit 673062fe62cacc663258855efa1333225e605698
Merge: 3c53ac7 e97c03c
Author: Hosanna <miramonte23@gmail.com>
Date: Mon Nov 2 11:21:16 2015 -0500

   Merge branch 'parser' of https://github.com/adamincera/dots into parser

commit 3c53ac74db4e3538b503f54f1bd3e7398ec5bdb0
Author: Hosanna <miramonte23@gmail.com>
Date: Mon Nov 2 11:20:52 2015 -0500

   compiler and the C code equiv to byte code

commit e97c03cebeae4a5378c98f44306537a2e322faac
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 2 10:52:29 2015 -0500

   added syntax for && and || symbols. fixed FOR and WHILE loop syntax

commit 1492a8856137a1f8ac5a79e9edb07716c1a8ecad
Author: rgordon <rcgordon@umass.edu>
Date: Mon Nov 2 10:24:28 2015 -0500

   fixed syntax of FOR and WHILE loops. addressed all todo comments. added
      rule comments to the expr definition in ast.ml. fixed dict declaration
      assignment (now accepts id { id : expr, id : expr.... ) etc.

commit 7059eccd39eab9075dadd63ea8b7e1e990abb428
```

```
Merge: df03698 bb0ebb4
Author: Hosanna <miramonte23@gmail.com>
Date: Mon Nov 2 10:16:07 2015 -0500

    Merge branch ’parser’ of https://github.com/adamincera/dots into parser

commit bb0ebb49bc4701ab9e58d6973f954245d59c232a
Author: Yumeng Liao <yl2908@columbia.edu>
Date: Mon Nov 2 02:45:15 2015 -0500

    after setting up VM and testing, fixed so testing system actually works now
        ...

commit 16930f181d84a284adf3f633f9724c661cc9d663
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Nov 1 23:39:58 2015 -0500

    Wrote python shell script maker that combines tests from all 4 txt files
        into tests.sh to be run. See instructions inside .txt for acceptable
        input. Could possibly be used in other tests later.

commit 4ed51c78e3607f83709cad061a50a1f06d7023d8
Merge: fa153c1 b7d616d
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Nov 1 22:58:16 2015 -0500

    Merge branch ’parser’ of https://github.com/adamincera/dots into parser

commit fa153c19b5d438d0bd976c0775bae79bf431a79a
Merge: bce2ffe e9cea73
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Nov 1 22:57:39 2015 -0500

    Merge branch ’master’ into parser

commit b7d616d2a94bf2952f6c8fe11b1567a447b5c1b7
Author: rgordon <rcgordon@umass.edu>
Date: Sat Oct 31 11:52:23 2015 -0400

    fixed dict and list declarations to require a *data_type* inside the type
        indicator (before it required ID’s, which is wrong). also added member
        access for lists and dicts to the expr cateogry (i.e. random access for
         dicts and lists via brackets)

commit df03698c0e146fca61e72e9c5e75a87b9f2c74a7
Merge: 9d5cd99 2431ac5
Author: Hosanna <miramonte23@gmail.com>
Date: Wed Oct 28 18:37:27 2015 -0400

    Merge branch ’master’ of https://github.com/adamincera/dots into parser

commit d455d5b5effabb5502661e7e88ea36886d989659
Merge: ac3904c 2431ac5
Author: rgordon <rcgordon@umass.edu>
```

```
Date: Wed Oct 28 18:32:41 2015 -0400

   Merge branch 'master' into parser

   merging change from master into parser branch. (updated lrm files)

commit ac3904cccab6634da34eb291323eb1b926b5c187
Author: rgordon <rcgordon@umass.edu>
Date: Tue Oct 27 12:19:40 2015 -0400

   added rules for declaring lists and dicts. what goes within [ ] or { } for
      assignment still needs to be reworked. also added folder of text files
      of token sequences that should be accepted in menhir

commit 9d5cd99922da2b7f196d98b95c03fb88e443a213
Merge: 1dbdcbd fe6a0b7
Author: Hosanna <miramonte23@gmail.com>
Date: Tue Oct 27 11:19:18 2015 -0400

   Merge branch 'parser' of https://github.com/adamincera/dots into parser

commit fe6a0b7aad6aa626d13dcb673db2e86498a47dd6
Author: rgordon <rcgordon@umass.edu>
Date: Tue Oct 27 11:18:43 2015 -0400

   fixed problem where program was never accepted

commit 2431ac59cefad1ecf2590abe9b2808685a2b6b07
Author: rgordon <rcgordon@umass.edu>
Date: Mon Oct 26 07:54:01 2015 -0400

   additional edits

commit e9cea73d337580171ea5ff7f7239d5fe9f2630b8
Author: Adam Incera <aji2112@columbia.edu>
Date: Mon Oct 26 00:24:16 2015 -0400

   cleaned up LRM, added logo to cover page

commit bce2ffe2076e070eb3ad9be1c3e210b3f6d0c3a7
Author: rgordon <rcgordon@umass.edu>
Date: Thu Oct 22 14:11:03 2015 -0400

   old versions of all files were based on outdated version of microc compiler
      . updated each file with new version of microc compiler. added function
       for concatenating lists and altered the program rule to use that since
       all variable declarations now return lists.

commit 77d47745262d1f0e1b719ee1f5847894d4616631
Author: rgordon <rcgordon@umass.edu>
Date: Thu Oct 22 10:18:19 2015 -0400

   changed definition of program symbol to be an object with a variable list,
      function list, and command list (aka stmt list). added the '.' token to
```

```
        the scanner and added rules for called a member variable or member
       function of an object to the expr rule

commit 327b31c16e686fd804abab45b788ef19f990638e
Author: rgordon <rcgordon@umass.edu>
Date: Thu Oct 22 09:26:19 2015 -0400

   previous fix forgot to change the patterns of the prefix definitions to
       refer to itself --> corrected. also fixed chaining of graph
       declarations

commit 1d0ac252d61979690ee196f67fef2e1bad7868bb
Author: rgordon <rcgordon@umass.edu>
Date: Thu Oct 22 09:23:40 2015 -0400

   fixed chaining declarations of primitives and nodes

commit 53c49b72410109ceef3ed6cba051910106764156
Author: rgordon <rcgordon@umass.edu>
Date: Thu Oct 22 00:11:50 2015 -0400

   minor edits

commit 1dbdcbd96053cbe89328ace904feaa1f2b99efa5
Author: rgordon <rcgordon@umass.edu>
Date: Thu Oct 22 00:01:26 2015 -0400

   minor edits

commit 817cc418b1994be088fffc9c3c3403bed4c1d220
Merge: 121d493 246b3a0
Author: Adam Incera <aji2112@columbia.edu>
Date: Wed Oct 21 23:50:33 2015 -0400

   merged, removed pdf from repo

commit 121d493e3496111690be4d8bc2e34326e846aaaf
Author: Adam Incera <aji2112@columbia.edu>
Date: Wed Oct 21 23:48:24 2015 -0400

   added section 1 and section 2.2 to LRM

commit 246b3a06f87df1641c25b79bf3cae23f5c00e4e7
Merge: fd1fbff e965c02
Author: Hosanna <miramonte23@gmail.com>
Date: Wed Oct 21 23:43:02 2015 -0400

   Merge branch ’master’ of https://github.com/adamincera/dots

commit fd1fbff6b0b33aa68f41e1e4e25458c39bffb823
Author: Hosanna <miramonte23@gmail.com>
Date: Wed Oct 21 23:42:42 2015 -0400

   hosanna
   's contriubtion

commit e965c028d19b707bcd5f42b5f980c2e3daafc77e
Author: rgordon <rcgordon@umass.edu>
Date: Wed Oct 21 23:41:38 2015 -0400

   added built-in function subsection

commit 3ffcd8ad51551cf2f850a5eecef59465a7bc66be
Author: rgordon <rcgordon@umass.edu>
Date: Wed Oct 21 22:10:10 2015 -0400

   slight change to name of rule

commit 61a87f635d17ab5ff8768ffeba5d610c2917df7f
Author: rgordon <rcgordon@umass.edu>
Date: Wed Oct 21 21:38:32 2015 -0400

   additional changes; added rules for edge operations

commit 5db14b42a4ece367a051e8954093572d0b4615fb
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Wed Oct 21 21:07:25 2015 -0400

   added right graph

commit d716ac0736f352a4b71944f6a7c4f84a7d851022
Author: rgordon <rcgordon@umass.edu>
Date: Wed Oct 21 21:01:08 2015 -0400

   1st attempts at making scanner and parser for our language. for testing
      whether the grammar is unambiguous

commit 7f980d206d07c1f0bc6faf1bce6684efc3cdefa4
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Mon Oct 19 21:25:19 2015 -0400

   Finished section on statements.

commit 7f2ee8c9ae79ab53a71202161deb2e4f42031389
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Mon Oct 19 20:18:38 2015 -0400

   added whitespace section, small tweaks to scoping

commit 3556afc508aa4b733ff08cee243b4416113619d3
Author: rgordon <rcgordon@umass.edu>
Date: Mon Oct 19 00:54:40 2015 -0400

   added sections for list and dict data types

commit 4c462efa38de3182801882aca408fd799056d35d
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Oct 18 17:39:43 2015 -0400
```

```

    Tweaked "scope" more to my liking with a more comprehensive definition.

commit c51c53eaf772bbd049e18b07d37c5d1f13425751
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Sun Oct 18 17:15:25 2015 -0400

    added scope section in Latex

commit ed5c07efdb001cd3c6d81faec72fcb74d8b10799
Author: rgordon <rcgordon@umass.edu>
Date: Sun Oct 18 14:54:50 2015 -0400

    created initial latex file for the language reference manual, including
        adding necessary headers and package imports. created Makefile for
        easily compiling and cleaning up the tex file. added primitive type,
        functions, and program structure sections to the doc.

commit 0846e4b9748002a74f1bbc7219ec59d98c88cbfe
Author: Adam Incera <aji2112@columbia.edu>
Date: Thu Oct 8 20:19:31 2015 -0400

    moved proposal into documentation directory, added latex compilation script

commit a6b182e60d3366ee7cfd9f20ded2dc340a3422be
Merge: 36f25c8 0e4350a
Author: Hosanna <miramonte23@gmail.com>
Date: Thu Oct 8 20:00:13 2015 -0400

    hosanna was here

commit 36f25c8db74c8497ce0cc4774dd616209d4b4c3f
Author: Hosanna <miramonte23@gmail.com>
Date: Thu Oct 8 19:59:02 2015 -0400

    hosanna was here

commit 8fadefb68aa67f9d9ef976b3074a51794415285e
Author: rgord <rcg2130@columbia.edu>
Date: Fri Oct 9 07:46:04 2015 -0400

    transferred latex project files for lang proposal from sharelatex to repo

commit 0e4350a21eb3b9eafe27dde4b3b3811bdf21cec9
Author: Yumeng Liao <y.liao.2908@gmail.com>
Date: Thu Oct 8 19:57:30 2015 -0400

    testing

commit 2306aed112353a1dd50308f26cdd6c83de7cbbff
Author: Adam Incera <aji2112@columbia.edu>
Date: Sun Sep 20 21:34:23 2015 -0400

```

```
    initial commit
```

Listing 41: git log