Twister: Final Report

Manager: Anand Sundaram (as5209)
Language Guru: Arushi Gupta (ag3309)
System Architect: Annalise Mariottini (aim2120)
Tester: Chuan Tian (ct2698)

May 11, 2017

Contents

1 Introduction

Twister is a language designed for matrix manipulation with a focus on applications to image processing. Twister allows creation, access to, and modification of 2-dimensional matrices of integers, floating point numbers, or tuples of either of these scalar types, effectively allowing up to 3-dimensional matrices. It is an imperative programming language with static scoping and strong typing.

An example application of Twister is to compute the convolution of two matrices for image processing applications. We show how this is implemented in our language.

2 Language Tutorial

Twister is a simple scripting language that supports variable-size two-dimensional matrices, as well as variable-size lists and tuples, which are both one-dimensional arrays with the restriction that tuples may only contain integers or floating point numbers. Twister also supports nested function declarations and passing around these complex datatypes by reference to allow modification of matrices.

Suppose we want to write a function that scales a matrix by a variable amount. We could start with an empty function declaration with the appropriate type and syntax:

```
fun scale_mat = (m: Matrix<int>, multiplier: int) -> Matrix<int> {

};
```

Notice that this empty function is not a valid function in Twister, as we do not return a value from it. Here, the following syntax indicates that we are declaring a new function named **fname** with the function signature and execution instructions specified in **fdecl**:

```
fun <fname> = <fdecl>
```

The following syntax at the beginning of the function declaration indicates the function being declared takes two arguments, a Matrix containing integer elements and an integer, and returns a Matrix with integer elements:

```
(m: Matrix<int>, multiplier: int) -> Matrix<int>
```

Finally, the instructions which specify what the function does are enclosed between curly braces, and a semicolon indicates the end of the declaration statement.

Now, to actually accomplish the scaling, we would need to dynamically determine the matrix dimensions, and create two new for loops that iterate over the indices within those dimensions:

```
fun scale_mat = (m: Matrix<int>, multiplier: int) -> Matrix<int> {
    for (row in range(0, m.num_rows)) {
        for (col in range(0, m.num_cols)) {

        }
    }
};
```

In this code, the following syntax is a special Twister language feature that allows iterating over all possible values of the variable named **loopvar**, which is implicitly assumed to be an integer, starting from **startval** and ending with the largest integer below **endval**:

```
for (loopvar in range(startval, endval))
```

Furthermore, the attributes **m.rows** and **m.cols** are ways of accessing the dimensions of the matrix **m**.

Finally, we just need to update the values in the matrix and return the result:

```
fun scale_mat = (m: Matrix<int,; multiplier: int) -> Matrix<int> {
    for (row in range(0, m.num_rows)) {
        for (col in range(0, m.num_cols)) {
            m[row][col] = m[row][col] * multiplier;
        }
    }
    return m;
};
```

This brief tutorial shows how to construct a function with simple functionality using Twister's syntax. The complete Language Reference Manual below provides explicit details about all of Twister's language features.

3 Language Reference Manual

3.1 Comments

- Single-line — any characters following the string // are ignored until a newline is encountered
- Multi-line — any characters after the string /* are ignored until the string */ is encountered

3.2 Identifiers

Variable names must match the following regex:

```
['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']*
```

3.3 Keywords

The following words are keywords and cannot be used as variable names: int, float, bool, char, fun, List, String, Tup, Struct, Matrix, if, else, return, and, or, not, for, fread, fwrite, print, print_int, println_int, println, range, toint, tofl, True, False.

3.4 Initializing and Declarations

Initialization and declaration must occur in the same statement. Variables are declared by specifying the variable type and the identifier separated by a space and initialized by an assignment operator that takes the right hand side of the assignment operator and stores it in the specified operator.

```
<type> <id> = <expr>
```

Where `<type>` is either a primitive or complex type, `<id>` is a valid identifier, and `<expr>` is an expression that returns type `<type>`.

3.5 Primitive Types

3.5.1 Scalar primitive types

Type	Description	Initialization Example
int	an integer	`int a = 5`
float	a floating point number	`float a = 5.2`

Scalar constants: Literal strings of digits are interpreted as integers; strings of digits containing one interior "." character are interpreted as floats. Both of these are scalar constants and can be used to assign specific values to scalar variables when declaring them, as shown in the "Initialization Example" column of the table above.

3.5.2 Other primitive types

Type	Description	Initialization Example
bool	a boolean	`bool x = True`
char	a character	`char x = ’a’`

Constants: Any single character enclosed in single quotes is interpreted as a character constant. The keywords True and False are interpreted as Boolean constants. These constants can be used to assign specific values when declaring the corresponding type of variable; examples are shown in the table above.

3.6 Complex Types

3.6.1 Summary

Type	Description	Attributes	Initialization Example
List<elem_type>	a mutable array	[i], len	`List<char> l = {’a’, ’b’,’`
String	a List of chars	[i], length	`String s = ’hello’`
Tup<elem_type>	an n-tuple of value	[i], len	`Tup<float> t = (0.0, 3.1,`
Matrix<elem_type>	a series of scalar arrays	[i][j] num_rows, num_cols	`Matrix<int> m = [0,0,0;0,0`
fun	a function (input to output)	—	`fun x = (i:  int) -> int {`

3.6.2 Lists

Initialization: Lists are initialized using List literals, which consist of two curly braces with elements separated by commas. An empty list may be initialized with a pair of braces with no elements.

```
List empty = {};
List stuff = {1,2,3};
```

Element Access/Assignment:

- [i]
  - As expression: accesses position i of a List
  - As assignment: assigns to position i

```
List<char> l = {’a’, ’b’, ’c’, ’d’};
l[2] = l[1]; // L = {’a’, ’b’, ’b’, ’d’}
```

Other Attributes:

- Length - the length of a List may be accessed with the ".len" attribute

3.6.3 Strings

Description: Strings are sequences of characters that can be printed but not indexed into or combined in any way.
Initialization: A String literal consists of a pair of doubles quotes surrounding zero or more chars.

```
String s = "abc";
```

3.6.4 Tuples

Description: An n-Tuple (aka Tuple) is a collection of n-many identically typed values. The value n of a Tuple cannot change after initialization.
Initialization:: A Tuple is initialized with a pair of parentheses that surround n-many expressions that return the same type.

```
Tup<int> t = (1, 2, 2 + 4);
```

Attributes:

- $[i]$
  - As expression: returns the element at position i of the Tuple
  - As assignment: assigns an element of the same type to position i of the Tuple
- Length - the attribute ".len" returns the value n of an n-Tuple

3.6.5 Matrices

Description: Matrices are two dimensional collections of scalars or tuples of scalar types.
Literal Initialization: You may use a Matrix literal to initialize a Matrix object. A Matrix literal consists of a pair of square brackets with commas separating the elements of of a row, and semicolons designating the beginning of a new row.

```
Matrix<int> m = [1,2,3;4,5,6;7,8,9];
```

Attributes: Each of the following selectors can be used by using the identifier of a Matrix object followed by the selector name.

- Element access
  - $[i][j]$ — returns the element in row i and column j.
- Element assignment
  - $[i][j]$ — assigns to the element in row i and column j.
- Row/Column Lengths — the ".num_rows" and ".num_cols" attributes return the number of rows and number of columns, respectively, of the matrix.

Demonstration of Attribute Usage:

```
Matrix<int> a= [0,0,0;0,0,0];
int r = a.num_rows; // 2
int c = a.num_cols; // 3
a[0][0] = 3; // a := [3, 0, 0; 0, 0, 0]
a[0][0] = a[0][0] + 3;
```

3.6.6 Functions

Initialization: A function is initialized in the following manner:

```
fun x = (<id>: <type-1) -> <type-r> {
<BODY>
    return <expr>;
};
```

Where `<id>` is a valid identifier, `<type-*>` is a primitive or object type, `<BODY>` is a series of statements, and `<expr>` is an expression that returns type `<type-r>`. To pass in addition arguments, you may add in other `<id>: <type>` pairs, sequenced by commas.
Function Calls: A function is called by its id, followed by a pair of parentheses with 0 or more expressions (to be passed as arguments) separated by commas.

```
fun sum = (x: int, y: int) -> int {
    return x + y;
};

fun sum_matrix = (m: Matrix<int>) -> int {
    int sum = 0;
    for (i in range(0, m.num_rows)) {
        for (j in range(0, m.num_cols)) {
            sum = sum + m[i][j];
        }
    }
    return sum;
};

Matrix<int> a = [1,2;3,4];

int b = sum_matrix(a);//returns 10
```

3.6.7 Object Typing

The List and Matrix objects have associated element types that are defined upon initialization; variables of these types must be declared with a specific element type. To manually specify this type, immediately after the object type use triangle brackets to contain the desired element type.

```
Matrix<int> M = [1,1;1,1]; // A matrix of integers
List<float> F = [1.5 , 2.5] //A list of floating point numbers
```

3.7 Operators

3.7.1 Arithmetic Operators

Operator	Description
$-$	the subtraction operator
$+$	the addition operator
$*$	the multiplication operator
/	the division operator
%%	the modulo operator

The arithmetic operators may be used between two scalar values or one scalar and one matrix to perform scalar multiplication as expected.

```
int x = 3;
int y = 4;
int z = x + y; // 7
```

3.7.2 Logical Operators

Operator	Description	Examples
==	equality comparison operator	`1 == 1 // True`
>	greater than operator	`1 > 2 // False`
<	less than operator	`1 < 2 // True`
>=	greater than or equal to operator	`1 > 2 // False`
<=	less than or equal to operator	`1 < 2 // True`
and	takes two bools and returns their AND	`(1 > 2) and (2 == 2) // False`
or	takes two bools and returns their OR	`(1 > 2) or (2 == 1) // True`
xor	takes two bools and returns their XOR	`(True xor False) // True`
not	returns the negation of a boolean	`not (1 == 1) // False`

3.8 Flow Control

3.8.1 For Loops

A 'for' loop is defined as an iteration over a List or Matrix with a named variable to represent the current element. If looped over a List, the elements iterate in indexed order. If looped over a Matrix, the elements iterate over rows first, then columns, in indexed order.

```
for(<var> in <values>) {
    <BODY>
}
```

Where `<var>` is any type, `<values>` is a List of values of that type, and `<BODY>` contains any sequence of statements that are valid Twister code.

```
for(elem in range(0,3)) {
    print(elem);
} // output:
// 0
// 1
// 2
```

3.8.2 If-Else Statements

Users can check booleans and conditionals with if statements, with an optional else block to run if the condition is not satisfied.

```
int a = 5;
if (a == 5) {
    int b = print("a was 5");
} else {
   int b = print("a was not five");
}
```

The exact syntax of the if block is

```
if (<CONDITION>) {
<BODY-1>tests
} else {
<BODY-2>
}
```

Where `<CONDITION>` is any expression that evaluates to a Boolean value, and `<BODY-1>` and `<BODY-2>` contain any sequence of statements that are valid Twister code.

3.8.3 Iterating over matrices

3.9 Built-in functions

Function	Description
print	takes a String and a file and writes the string to the file
println	takes a String and a file and writes the string to the file + a newline
print_fl	takes a float and writes it
print_c	takes a char and writes it
print_int	takes an int and writes it
println_int	takes an int and writes it to file + a newline
range	takes two arguments a and b, returns a List of ints = $\{a, a+1, ..., b-1\}$
tofl	converts ints to floats
toint	converts floats to ints

3.10 Example Program

```
fun small_c = ( i: int, j: int, img: Matrix<int>,  kernel : Matrix<int>) -> int {
int nr = kernel.num_rows;
int nc = kernel.num_cols;

int endrow  = i + nc;
int endcol = j + nr;
int sum = 0;
for (mr in range(i, endrow)) {
    for (mc in range(j, endcol)) {
    int imen  = img[mr][mc];
    int keren = kernel[mr-i][mc-j];
    sum = sum + keren*imen;
    }
}

return sum;
};

Matrix<int> res = [0, 0, 0; 0, 0, 0; 0, 0, 0];

fun convol = (img: Matrix<int>, kernel : Matrix<int>) -> Matrix<int> {
    int imw = img.num_cols;
    int imh = img.num_rows;
    int knw = kernel.num_rows;
    int redw = imw - knw + 1;
    int redh = imh - knw + 1;

    for (i in range(0, redh)) {

        for (j in range(0, redw)) {
            res[i][j] =  small_c(i, j, img, kernel);
        }
    }

    return img;
};


Matrix<int> cimg = [ 18, 2 , 0, 1, 5; 3, 1, 2, 1, 4; 7,9,3,1,2; 9,5,6,7,1; 0,8,6,2,3];
Matrix<int> cker = [ 1, 2, 1; 2, 3, 1; 4, 5, 6];
int x = 0;
int y= 0;
int b = small_c( x,y, cimg, cker);
Matrix<int> ik = convol(cimg, cker);


for ( d in range(0, 3)) {
    for (q in range(0, 3)) {
        int j = print_int(res[d][q]);
        if (q == 2) {
            j = println(" ");}
        else{
            j = print(" ");
        }
    }
}
```

3.11 Scope

Any variable defined between braces, {}, has a local scope within those braces and goes out of scope when the code between those braces finishes executing. Otherwise, a variable's scope extends from the time it is declared to the end of the program run. Variable declared in loops do not have scope outside of the loop that they are declared in. Variable names used for function arguments also have scope limited to the function they are defined in.

3.12 Compilation

Twister compiles down to LLVM.

4 Project Plan and Implementation

4.1 Members

Manager: Anand Sundaram (as5209)
Language Guru: Arushi Gupta (ag3309)
System Architect: Annalise Mariottini (aim2120)
Tester: Chuan Tian (ct2698)

4.2 Version Control

```
Branches Summary

Master

Contains code that has been reviewed and tested.
Should NEVER be pushed to or committed to directly.
If you want to make a change to master, make a new branch, make your changes, and submit a PR (see
QA

Contains code to be tested before it gets merged into master.
Not guaranteed to be functional.
Shall be re-synced with master every Sunday at 11:59 PM. If you have a PR up while this re-sync oc
Contributing to the Repo

Making a New Branch

All new branches should be branched off of master, and should follow the following naming format:

git checkout master
git pull origin master
git checkout -b yourname/branch-name
Committing

Commit early and commit often. Use descriptive commit messages that describe changes made.

GOOD: git commit -m "expanded identifier type in scanner"

BAD: git commit -m "scanner"

Keeping Commits Clean

Scenario 1: you just committed, but realize you had more you wanted to add to that commit

Run the following command to add your changes to the previous commit:
git commit --amend --no-edit
Scenario 2: you have many redundant commits in your commit history

Run the following command (this example is for changes up to 3 commits back)
git rebase -i HEAD~3
A vi editor will open with your commit log, which should look something like this:
pick f392171 Added new feature X
pick ba9dd9a Added more stuff
pick df71a27 Oops I broke something, fixed it
For any commits you want to squash, change "pick" to "squash" (note: commits squash up), as follow
pick f392171 Added new feature X
squash ba9dd9a Added more stuff
squash df71a27 Oops I broke something, fixed it
```

Finally, type ":wq" to exit the editor and let the rebase do its thing.
Making a Pull Request

When you want to merge a feature branch into master, you must create a pull request. For now, PRs

Perform the following steps for each PR:

Write a description that summarizes the new features in your branch.
Immediately resolve all merge conflicts. Branches with merge conflicts will not be reviewed.
Assign 2 other group members to approve the PR (under the "Reviewers" feature).
Checkout QA and merge in your branch.

```
git checkout qa
git pull origin qa
git merge yourname/branch-name
```

If your reviewers request changes, address those changes, then re-push to the branch and to QA. Wh

Reviewing a Pull Request

If you are assigned to review a PR, you must either approve or request changes within one week of

```
git checkout qa
git pull origin qa
bin/test.sh
```

4.3 Project Structure

File Names
File names should be lowercase_and_snake_case, unless convention or necessity dictates otherwise (

Directory Structure
The project will have 2 main directories.
The /src directory will contain the source code for the compiler, including the Makefile and all t

The /test directory will contain tests organized by compiler component and test type. There will b

4.4 Programming Style

Programming Style Guide
Indents should consist of 2 or 4 spaces (in either case, be consistent throughout the same file).

Line lengths should not extend beyond 120 spaces (approx the length of the GitHub file view).

Use descriptive variable names. Single character variables should only ever be used in innermost s

Any commented TODOs should take the following form: TODO description date initials, e.g.

```
// TODO write description of todo 2/20/17 AIM
```

OCaml Specific Style Guide

Follow variable naming conventions as given in class lectures. Programs should resemble the style

Avoid imperative elements as much as possible ("don’t smoke!"). If you believe an imperative eleme
Top-level elements should all have a comment giving a short description of their purpose. Large to

Files should not exceed 500 lines in length. If your file exceeds this length, break your file up
Use ;; at the end of every complete statement, even if technically unnecessary.
Use whitespace liberally. Use spaces after commas, around operators, and to align elements.

The keyword "in" should occur at the end of a single line "let" statement, or on it’s own line if

```
Files should not contain print statements unless there's a really really good reason for it.

Some Examples:

GOOD:

(* global variables* )
let literal_one = 1;;
let literal_two = 2;;
BAD:

let x = 1
let y = 2
GOOD:

match x with
LONGLONGLONGNAME -> x + 1
| SHRT            -> x + 0
BAD:

match x with
LONGLONGLONGNAME -> x+1
| SHRT -> x+0
GOOD:

(* this function is here for demonstration purposes *)
let num = 1 in
    let add x =
        print_endline "I'm printing here for a good reason";
        num + x
    in
    add 4;;
BAD:

let a = 1
    in let add x =
        print_endline "I was using this to debug and forgot to remove it";
        a+x in
    add 4;;
```

4.5 Project Timeline

```
Feb 22  Scanner compiles! Seems mostly complete.
Mar 1   Parser, ast, and scanner all build w/o errors!
Mar 20 Added a codegen file
Mar 25 Hello world works now
Mar 26 Tests running on Travis
Apr 24 Semant is ready for testing!!
May 8  Matrix assignment works in codegen
May 10  Final presentation
```

4.6 Project Log

```
Commits on May 10, 2017
 @anandsun
Minor test changes
anandsun committed 39 minutes ago
```

```
17d2745
 @anandsun
Minor changes to to_string; rebuild semant
anandsun committed an hour ago
ba01244
 @anandsun
Revert "Merge in graft"  ...
anandsun committed 2 hours ago
29db08d
 @anandsun
Merge in graft
anandsun committed 2 hours ago
04593dd
 @anandsun
Some more cleanup
anandsun committed 2 hours ago
721369d
 @anandsun
Fix type disambiguation errors
anandsun committed 3 hours ago
84433e7
 @anandsun
Boolean operations work
anandsun committed 3 hours ago
d6fef9f
 @anandsun
Fixed floating point math
anandsun committed 4 hours ago
7081717
 @anandsun
Remove unsupported stuff
anandsun committed 4 hours ago
466433b
Commits on May 9, 2017

nested for loops work
agup committed 18 hours ago
16dd762

fixed bug in order arguments were evaluated
agup committed 19 hours ago
1504dff

fixed outer level, added matrices to funcntion signature, fixed way m... ...
agup committed 19 hours ago
fd8b5b3

added more cases to additional argument list
agup committed a day ago
0737d7e
 @anandsun
Fix assignments!
anandsun committed a day ago
b6e9953

added more cases to outer level functions
agup committed a day ago
b7c5c59
```

```
 @anandsun
Use decls instead of assigns for printing
anandsun committed a day ago
b5e22cd
 @anandsun
Graft working except for assignment
anandsun committed 2 days ago
14ee943

fixed bad naming in if which caused it to break
agup committed 2 days ago
55710cb
Commits on May 8, 2017
 @anandsun
Troubleshooting if/else
anandsun committed 2 days ago
c86852a

Merge branch 'anand/graft-branch' of https://github.com/TianchuanGitH...  ...
agup committed 2 days ago
fde839c

fixed matrix lhs assignment in codegen
agup committed 2 days ago
2e2563a
 @anandsun
Allow matrix initialization; change types from scalar literals to exp...  ...
anandsun committed 2 days ago
eb87a7c
 @anandsun
Merge in records from record-branch  ...
anandsun committed 2 days ago
5359f1c
 @anandsun
Broke out scalar binops; binops can now check type
anandsun committed 2 days ago
3d53b79
 @anandsun
Eval returns Ast type recursively!
anandsun committed 2 days ago
c5c6937
 @anandsun
All record-like
anandsun committed 2 days ago
ab4d5b7

matrix assignment works in codegen
agup committed 2 days ago
c17b130
 @anandsun
Use records instead of tuples - part 1
anandsun committed 2 days ago
6b5b426
Commits on May 7, 2017
 @anandsun
Matrix index assignment should work
anandsun committed 3 days ago
1480816
```

```
 @anandsun
Add lli to shell script; prevent matrices from being indexed with 1 i...  ...
anandsun committed 3 days ago
d83e14f
 @anandsun
Fix warnings in semant, turn semant on, fix shell script for testing ...  ...
anandsun committed 3 days ago
0117c8e
 @anandsun
Fix warnings in to_string and semant
anandsun committed 3 days ago
141610f
 @anandsun
Realigning
anandsun committed 3 days ago
de64996
 @anandsun
Get parser and semant to support matrix on LHS
anandsun committed 3 days ago
954604f

remove useless code
agup committed 3 days ago
60ba7ae

Commits on May 7, 2017

Merge branch 'arushi/codegen' of https://github.com/TianchuanGitHub/P...  ...
agup committed 3 days ago
68bc351

working on append
agup committed 3 days ago
d0f6129
 @anandsun
Refactor array syntax and matrix/list type calculation
anandsun committed 3 days ago
086daa5
 @anandsun
Revert "Merge in for loops"  ...
anandsun committed 3 days ago
920656c
 @anandsun
Merge in for loops
anandsun committed 3 days ago
4a8aeb3

for loops over ranges and lists now work
agup committed 3 days ago
40f3182

added additional arguments to eval
agup committed 3 days ago
bfe7f43

less cases in print function
agup committed 3 days ago
f84ec14
```

```
Merge branch 'arushi/codegen' of https://github.com/TianchuanGitHub/P...  ...
agup committed 3 days ago
2e6ea8d

added ast types to variable maps
agup committed 3 days ago
704f114
 @anandsun
Minor refactoring
anandsun committed 3 days ago
2432e9d
 @anandsun
Make A.MatIndex type source explicit; refactor semant binop type chec... ...
anandsun committed 3 days ago
8c7d4ab

added Anand's list fix for indexing
agup committed 3 days ago
410b0a9

added toint casting
agup committed 3 days ago
053f740

made printing cases more complete testing tofl now
agup committed 3 days ago
2df3c26

if loops should work outside function bodies
agup committed 3 days ago
ac9823d

changed outer level functions still working on if removed comments
agup committed 3 days ago
b14254d

fixed for loops number of iterations for lists
agup committed 3 days ago
1e1caf6
Commits on May 6, 2017

Merge branch 'anand/heap-arrays' into arushi/codegen
agup committed 4 days ago
0919761

prep for merge
agup committed 4 days ago
85e7797
 @anandsun
Get heap-style arrays working
anandsun committed 4 days ago
4cf77f1
 @anandsun
Merge in printing
anandsun committed 4 days ago
7ea1e81
 @anandsun
```

```
Heapwise list declaration and access compiles
anandsun committed 4 days ago
a7652a9

Merge branch 'arushi/codegen' of https://github.com/TianchuanGitHub/P...  ...
agup committed 4 days ago
cee5ed8

added printing functionnality inside functions
agup committed 4 days ago
5825b2c
 @anandsun
Revert "Merge in for loops"  ...
anandsun committed 4 days ago
4ec7243
 @anandsun
Merge in for loops
anandsun committed 4 days ago
d799bc8
Commits on May 5, 2017

adding test script
Chuan Tian committed 5 days ago
47e3c39

added list access code and finished up the for loop
agup committed 5 days ago
0c98f6e

made lists back to the way they were on the stack
agup committed 5 days ago
421c73d
 @anandsun
Fix compile error
anandsun committed 5 days ago
7233e03
 @anandsun
Only check local scope for var names
anandsun committed 5 days ago
5be0267

added back changes that were erased during merge
agup committed 5 days ago
ae68cda
 @anandsun
FIxes to binop evaluation, printing of declarations, list access  ...
anandsun committed 6 days ago
4c2dd99
Commits on May 4, 2017
 @anandsun
Fix bools in parser, scanner; try to add pointer types for Matrix
anandsun committed 6 days ago
06eeadd

Commits on May 4, 2017
 @anandsun
Pull for loops
anandsun committed 6 days ago
```

```
a444687

Merge branch 'arushi/codegen' of https://github.com/TianchuanGitHub/P...  ...
agup committed 6 days ago
c970aa5

for loops work in the sense that they iterate teh correct number of t... ...
agup committed 6 days ago
271bc4f
Commits on May 3, 2017
 @anandsun
Matrix access tests
anandsun committed 7 days ago
d87dff2

Anand's matrix initialization
agup committed 7 days ago
0eaa9bc

corrected if statemennt RBRACE syntax
agup committed 7 days ago
e58cb85

added more predicates to if statements
agup committed 7 days ago
ad8c996

if loops work now for > predicates
agup committed 7 days ago
1dc80e4
Commits on May 2, 2017
 @anandsun
Add list and tuple literals to expression evaluation; create codeine ...  ...
anandsun committed 8 days ago
d186574

Merge branch 'arushi/codegen' of https://github.com/TianchuanGitHub/P...  ...
agup committed 8 days ago
c6e72b1

added list initialization
agup committed 8 days ago
9e26be4
Commits on May 1, 2017
 @anandsun
Merge branch 'anand/semantic-checks' into arushi/codegen
anandsun committed 9 days ago
d803622
 @anandsun
Fixed bug in return type after currying; also fixed warnings
anandsun committed 9 days ago
e4cded8
 @anandsun
Finished last 2 built-in functions
anandsun committed 9 days ago
ab5038e
Commits on Apr 29, 2017
 @anandsun
```

```
Minor comment changes
anandsun committed 11 days ago
a39564b
 @anandsun
Merge branch 'anand/semantic-checks' into arushi/codegen  ...
anandsun committed 11 days ago
1abe53a
 @anandsun
Add built-in functions
anandsun committed 11 days ago
403b0ad
 @anandsun
Ignore secant binary!
anandsun committed 11 days ago
b266551
 @anandsun
Merge remote-tracking branch 'origin/anand/semantic-checks' into anan...  ...
anandsun committed 11 days ago
8b9ec4e
Commits on Apr 28, 2017

tried to add functionality for lists does not parse so cannot check
agup committed 12 days ago
e184a5a
 @anandsun
Allow empty lists, tuples, and matrices, and fix logic for checking i...  ...
anandsun committed 12 days ago
e7ac729

adding test
Chuan Tian committed 12 days ago
a207dbe
 @anandsun
Add matrix access checking to secant
anandsun committed 12 days ago
873e34e
 @anandsun
Make syntax consistent in ast
anandsun committed 12 days ago
d74064d
 @anandsun
Added matrix access to parser and art
anandsun committed 12 days ago
e5def7e

added tofl
agup committed 12 days ago
a7d4683
Commits on Apr 27, 2017

added binnops
agup committed 13 days ago
4e0fc79
 @anandsun
Edit alignments again
anandsun committed 13 days ago
687eaec
 @anandsun
```

```
Merge remote-tracking branch 'origin/arushi/codegen' into arushi/codegen  ...
anandsun committed 13 days ago
282be9a

function call broken but compiles
agup committed 13 days ago
3f2224e
 @anandsun
Re-edit alignments
anandsun committed 13 days ago
9e16971

formated using textedit
agup committed 13 days ago
ff12934

spacing used texedit
agup committed 13 days ago
ddcd1b7

fixed spacingn on nlast function
agup committed 13 days ago
0571b80

spaces
agup committed 13 days ago
493ced9

changed spacing again
agup committed 13 days ago
a3dfaa8

added comments
agup committed 13 days ago
7f27735

changed some spacing and added variable reassignment
agup committed 13 days ago
59bf270

added function calling
agup committed 13 days ago
355730b

Merge branch 'anand/semantic-checks' into arushi/codegen
agup committed 13 days ago
ccea8ec

made codegenn compatible with current parser
agup committed 13 days ago
f790228
Commits on Apr 25, 2017

adding new semant tests, all works well
Chuan Tian committed 16 days ago
d6a40ef
Commits on Apr 24, 2017
 @anandsun
```

```
Print full statement context when raising exception
anandsun committed 16 days ago
c577b26
 @anandsun
Fix first few bugs in semant  ...
anandsun committed 16 days ago
17ed4ab
 @anandsun
Fix spelling in comment
anandsun committed 16 days ago
559598d

adding semant tests
Chuan Tian committed 16 days ago
927cecf
 @anandsun
Fix local return type in function declaration
anandsun committed 16 days ago
92cedf6
 @anandsun
Fix quote compile error
anandsun committed 16 days ago
5736b5a
 @anandsun
Support currying and add newlines around pretty-printing complex expr...  ...
anandsun committed 16 days ago
04760de
 @anandsun
Rename test_parser to test_semant, put in skeleton of call to secant
anandsun committed 16 days ago
42b7a92

Merge branch 'anand/semantic-checks' of https://github.com/TianchuanG...  ...
Chuan Tian committed 16 days ago
a47a970

fixed scanner tests
Chuan Tian committed 16 days ago
4d7b53d
 @anandsun
Change global type to Bool
anandsun committed 16 days ago
83e8d9d
 @anandsun
Add comments (including TODOs) and prettify syntax
anandsun committed 16 days ago
bc44ee6
 @anandsun
Semant is ready for testing!!
anandsun committed 16 days ago
0dee114
 @anandsun
Semant compiles; parser handles multiple statements in if / for / fun... ...
anandsun committed 16 days ago
1dd72b9
Commits on Apr 20, 2017

travis#8, adding test for failing cases
```

```
Chuan Tian committed 20 days ago
ccceb74

travis#7, house keeping, adding failing cases
Chuan Tian committed 20 days ago
0d7cb09

travis#6 and house keeping
Chuan Tian committed 20 days ago
ae0c96f

travis#5 and house keeping
Chuan Tian committed 20 days ago
a8c7623

travis #4
Chuan Tian committed 20 days ago
da1d2b8

travis #3
Chuan Tian committed 20 days ago
70786ff

improving travis2
Chuan Tian committed 20 days ago
c4da0c8

improving travis
Chuan Tian committed 20 days ago
d6908ff
Commits on Apr 17, 2017
 @anandsun
Ignore Sublime
anandsun committed 23 days ago
4126dfd

fixed matrix
Chuan Tian committed 23 days ago
46015a0
Commits on Apr 13, 2017

accidentally broke scanner, now scanner give [invalid token] for '>' ... ...
Chuan Tian committed 27 days ago
4b51181

fixed an error, still working on it.
Chuan Tian committed 27 days ago
cc55f11

trying to fix the matrix prob, hope it won't break the parser
Chuan Tian committed 27 days ago
ee27eaa

parser tests are all passed, except fun3, which has problem with spec... ...
Chuan Tian committed 27 days ago
d218a88

testing
```

```
Chuan Tian committed 27 days ago
f60b64c

Merge branch 'parsing_branch' of https://github.com/TianchuanGitHub/P...  ...
Chuan Tian committed 27 days ago
d71bc5d
Commits on Apr 6, 2017
 @anandsun
Improve parser testing  ...
anandsun committed on Apr 6
67ecd13

scanner passed all tests
Chuan Tian committed on Apr 6
b2996c5
 @anandsun
Make function body multiple statements in AST and parser
anandsun committed on Apr 6
b585896
 @anandsun
Update pretty-printing for statement lists
anandsun committed on Apr 6
0458f80

adding stuff to test.sh
Chuan Tian committed on Apr 6
b669475

finished writing scanner tests
Chuan Tian committed on Apr 6
2302a37

check
Chuan Tian committed on Apr 6
26570a4
Commits on Apr 3, 2017
 @anandsun
Fix shift/reduce errors in EXPR; remove built-in functions as keywords
anandsun committed on Apr 3
b6f3975

adding parsing testing
Chuan Tian committed on Apr 3
8f917e7

Merge branch 'parsing_branch' of https://github.com/TianchuanGitHub/P...  ...
Chuan Tian committed on Apr 3
eadc9a9

testing travis
Chuan Tian committed on Apr 3
68bf063
Commits on Apr 1, 2017
 @anandsun
Fix gitignore
anandsun committed on Apr 1
e998a09
 @anandsun
```

```
Fix pretty-printing errors
anandsun committed on Apr 1
767bc9d
Commits on Mar 31, 2017

'adding base test, scanner still not do divide right
Chuan Tian committed on Mar 31
926c6c3
Commits on Mar 30, 2017

forking
Chuan Tian committed on Mar 30
192fcc3
Commits on Mar 28, 2017

Merge branch 'parsing_branch' of https://github.com/aim2120/PLT_project...  ...
Arushi Gupta committed on Mar 28
7093499

hello word print string works now
Arushi Gupta committed on Mar 28
0e8fb00
Commits on Mar 26, 2017

Tests running
Chuan Tian committed on Mar 26
845d3eb

clean up testing
Chuan Tian committed on Mar 26
Commits on Mar 31, 2017

adding base test, scanner still not do divide right"
Chuan Tian committed on Mar 31
b937c8c
Commits on Mar 30, 2017

forking
Chuan Tian committed on Mar 30
192fcc3
Commits on Mar 28, 2017

Merge branch 'parsing_branch' of https://github.com/aim2120/PLT_project...  ...
Arushi Gupta committed on Mar 28
7093499

hello word print string works now
Arushi Gupta committed on Mar 28
0e8fb00
Commits on Mar 26, 2017

Tests running
Chuan Tian committed on Mar 26
845d3eb

clean up testing
Chuan Tian committed on Mar 26
4833cb3
```

```
Commits on Mar 25, 2017
 @anandsun
Push To_String again
anandsun committed on Mar 25
 1   d11f3a2

hello world works now
Arushi Gupta committed on Mar 25
1b32e31

fixed error
Arushi Gupta committed on Mar 25
7d4c888

functions with return statements work now
Arushi Gupta committed on Mar 25
2e9494d

Merge branch 'parsing_branch' of https://github.com/aim2120/PLT_project...  ...
Arushi Gupta committed on Mar 25
c103258
 @anandsun
Add braces in function declaration
anandsun committed on Mar 25
b5bd3db

more function functionality
Arushi Gupta committed on Mar 25
025a6e2
 @TianchuanGitHub
Create .travis.yml
TianchuanGitHub committed on GitHub on Mar 25
fe9d863

Merge branch 'parsing_branch' of https://github.com/aim2120/PLT_project...  ...
Arushi Gupta committed on Mar 25
ff9a42b

added function functionality
Arushi Gupta committed on Mar 25
a9c67b4
 @anandsun
Make parser output verbose
anandsun committed on Mar 25
b299d4c

Merge branch 'parsing_branch' of https://github.com/aim2120/PLT_project...  ...
Arushi Gupta committed on Mar 25
3c22d8f

added functions to codegen
Arushi Gupta committed on Mar 25
4b74645
 @anandsun
Fixed if printing, removed makers
anandsun committed on Mar 25
9016653
 @anandsun
```

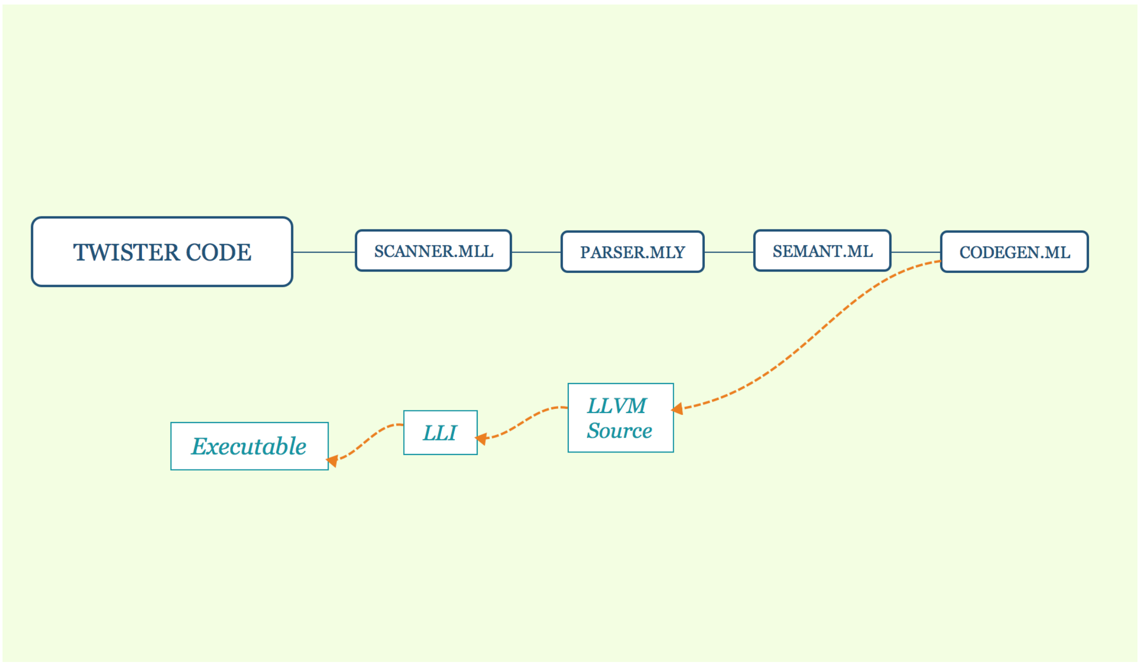
```
Fixed errors in compilation of to_string
anandsun committed on Mar 25
f44e059
 @TianchuanGitHub
Create trasvis.yml
TianchuanGitHub committed on GitHub on Mar 25
5216902
Commits on Mar 24, 2017

Merge branch 'parsing_branch' of https://github.com/aim2120/PLT_project...  ...
Arushi Gupta committed on Mar 24
9287c9e

added print fucntion
Arushi Gupta committed on Mar 24
c82e9ff

fun
Arushi Gupta committed on Mar 24
817368b

hardcoded some llvm ir code
Arushi Gupta committed on Mar 24
23fcd74
Commits on Mar 23, 2017
 @anandsun
Add project to ignore
anandsun committed on Mar 23
f0786ab
 @anandsun
Delete sublime stuff
anandsun committed on GitHub on Mar 23
417ad1c
 @anandsun
Delete sublime stuff
anandsun committed on GitHub on Mar 23
e88f0f6
 @anandsun
Ignore workspace
anandsun committed on Mar 23
d3413a8
 @anandsun
Fix make compile errors in to_string
anandsun committed on Mar 23
687c9da
 @anandsun
Clean up old files
anandsun committed on Mar 23
5277e11
 @anandsun
Remove arthop - these are binds
anandsun committed on Mar 23
 1   1e15d22
Commits on Mar 21, 2017
 @anandsun
Pretty printing
anandsun committed on Mar 21
e169cf6
```

```
Commits on Mar 20, 2017

fixed error message
Arushi Gupta committed on Mar 20

Commits on Feb 22, 2017
 @aim2120
Merge pull request #1 from aim2120/annalise/master-setup  ...
aim2120 committed on GitHub on Feb 22
2331663
Commits on Feb 20, 2017
 @aim2120
init src and test directories
aim2120 committed on Feb 20
177ef3b
 @aim2120
gitignore and clean script
aim2120 committed on Feb 20
8aac01b
 @TianchuanGitHub
Update README.md
TianchuanGitHub committed on GitHub on Feb 20
f1c63e8
 @TianchuanGitHub
Update README.md
TianchuanGitHub committed on GitHub on Feb 20
ab4d07b
 @TianchuanGitHub
Update README.md
TianchuanGitHub committed on GitHub on Feb 20
ffd31d3
Commits on Jan 23, 2017
 @TianchuanGitHub
Initial commit
TianchuanGitHub committed on Jan 23
fa429cd
```

4.7 Software Development Environment

We developed the project on our laptops. All of us but Annalise have Apple MacBooks, and she did not work on the parts of the compiler that involved llvm code generation, so we did not use a virtual machine, but instead just installed consistent versions of Ocaml and LLVM using Homebrew. We used OCaml version 4.04.0 and LLVM 3.9.1.

5 Architectural Design



Annalise Mariottini wrote the first version of scanner and parser.

Anand Sundaram did a lot to improve the scanner and parser.

Anand wrote the semant file to perform static semantic analysis and catch many types of syntax errors at compile type before attempting to generate llvm code.

Arushi Gupta wrote the majority of the code in the codegen file which generates LLVM IR code from semantically valid Twister programs. Anand contributed some features to the codegen.

Tian Chuan wrote tests for all the parts mentioned above.

6 Test Plan

6.1 Testing process

Test Procedures for scanner/ parser/ semant/ codegen: clean.sh build.sh test.sh Automating Tests for every build: .travis.yml .travis-ci.sh

We wrote our current test cases based on the questions and problems we encountered when we were implementing key functionalities of our language. Therefore, most of the test cases are related to matrix manipulation and calculation.

We were using Travis for automated Tests. Our compiler is tested every time we commit and pushed to our git-repository.

The tester is responsible for writing and implementing the tests. Other team members also contributed greatly to testing as well.

7 Test Examples

1. Boolean conditions and if/else statements

   Twister code:

```
int x = 3;
if (x < 4 and x > 3) {
   int l = print_int(15);
 } else {
   int l = print_int(22);
}
```

   LLVM code:

```
int x = 3;
if (x < 4 and x > 3) {
   int l = print_int(15);
 } else {
   int l = print_int(22);
}dyn-160-39-132-162:codegen anandsun$ cat cond_test.ll
; ModuleID = 'Twister'

@fmt = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt.1 = private unnamed_addr constant [3 x i8] c"%d\00"

declare i32 @printf(i8*, ...)

declare i32 @printbig(i32)

declare i8* @fopen(i8*, i8*)

declare i32 @fread(i8*, i32, i32, i8*)

define i32 @main() {
entry:
  %x = alloca i32
  store i32 3, i32* %x
  %"%tmp" = load i32, i32* %x
  %"%tmp1" = icmp slt i32 %"%tmp", 4
  %"%tmp2" = load i32, i32* %x
  %"%tmp3" = icmp sgt i32 %"%tmp2", 3
  %"%tmp4" = and i1 %"%tmp1", %"%tmp3"
  br i1 %"%tmp4", label %then, label %else

merge:                                            ; preds = %else, %then
  ret i32 0

then:                                             ; preds = %entry
  %print = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @fmt,
  %l = alloca i1
  store i1 true, i1* %l
  br label %merge

else:                                             ; preds = %entry
  %print5 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @fmt.
  %l6 = alloca i1
  store i1 true, i1* %l6
  br label %merge
}
```

2. List and Matrix types
   Twister code:

```
fun f = () -> int {
int q = 2;
int r = print_int(q);
r = print("\n");

List<int> y = {5,3,1};
int x = y[0];
r = print_int(x);

Matrix<int> bar = [6,18,7; 5,3,9];
int foo = bar[1][2];
int baz = print_int(foo);
return y[1];
};

int n = f();
```

LLVM code:

```
fun f = () -> int {
int q = 2;
int r = print_int(q);
r = print("\n");

List<int> y = {5,3,1};
int x = y[0];
r = print_int(x);

Matrix<int> bar = [6,18,7; 5,3,9];
int foo = bar[1][2];
int baz = print_int(foo);
return y[1];
};

int n = f();
dyn-160-39-132-162:codegen anandsun$ cat array_types.ll
; ModuleID = 'Twister'

@fmt = private unnamed_addr constant [3 x i8] c"%d\00"
@"%tmp" = private unnamed_addr constant [3 x i8] c"\5Cn\00"
@fmt.1 = private unnamed_addr constant [3 x i8] c"%s\00"
@fmt.2 = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt.3 = private unnamed_addr constant [3 x i8] c"%d\00"

declare i32 @printf(i8*, ...)

declare i32 @printbig(i32)

declare i8* @fopen(i8*, i8*)

declare i32 @fread(i8*, i32, i32, i8*)

define i32 @f() {
entry:
  %q = alloca i32
  store i32 2, i32* %q
  %"%tmp" = load i32, i32* %q
  %print = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @fmt,
  %r = alloca i1
  store i1 true, i1* %r
  %print1 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @fmt.
  store i1 true, i1* %r
  %malloccall = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* nu
  %"%tmp_list_ents" = bitcast i8* %malloccall to i32*
  %"%tmp_ptr" = getelementptr i32, i32* %"%tmp_list_ents", i32 0
  store i32 5, i32* %"%tmp_ptr"
  %"%tmp_ptr2" = getelementptr i32, i32* %"%tmp_list_ents", i32 1
  store i32 3, i32* %"%tmp_ptr2"
  %"%tmp_ptr3" = getelementptr i32, i32* %"%tmp_list_ents", i32 2
```

```
store i32 1, i32* %"%tmp_ptr3"
%malloccall.4 = tail call i8* @malloc(i32 ptrtoint (<{ i32, i32* }>* getelementptr (<{ i32,
%"%tmp_list_struct" = bitcast i8* %malloccall.4 to <{ i32, i32* }>*
%"%tmp5" = getelementptr inbounds <{ i32, i32* }>, <{ i32, i32* }>* %"%tmp_list_struct", i3
store i32 3, i32* %"%tmp5"
%"%tmp6" = getelementptr inbounds <{ i32, i32* }>, <{ i32, i32* }>* %"%tmp_list_struct", i3
store i32* %"%tmp_list_ents", i32** %"%tmp6"
%y = load <{ i32, i32* }>, <{ i32, i32* }>* %"%tmp_list_struct"
%y8 = alloca <{ i32, i32* }>
store <{ i32, i32* }> %y, <{ i32, i32* }>* %y8
%"%tmp9" = getelementptr inbounds <{ i32, i32* }>, <{ i32, i32* }>* %y8, i32 0, i32 1
%"%tmp10" = load i32*, i32** %"%tmp9"
%"%tmp11" = getelementptr i32, i32* %"%tmp10", i32 0
%x = load i32, i32* %"%tmp11"
%x13 = alloca i32
store i32 %x, i32* %x13
%"%tmp14" = load i32, i32* %x13
%print15 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @fmt
store i1 true, i1* %r
%malloccall.16 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32*
%"%tmp_mat_ents" = bitcast i8* %malloccall.16 to i32*
%"%tmp_ptr17" = getelementptr i32, i32* %"%tmp_mat_ents", i32 0
store i32 6, i32* %"%tmp_ptr17"
%"%tmp_ptr18" = getelementptr i32, i32* %"%tmp_mat_ents", i32 1
store i32 18, i32* %"%tmp_ptr18"
%"%tmp_ptr19" = getelementptr i32, i32* %"%tmp_mat_ents", i32 2
store i32 7, i32* %"%tmp_ptr19"
%"%tmp_ptr20" = getelementptr i32, i32* %"%tmp_mat_ents", i32 3
store i32 5, i32* %"%tmp_ptr20"
%"%tmp_ptr21" = getelementptr i32, i32* %"%tmp_mat_ents", i32 4
store i32 3, i32* %"%tmp_ptr21"
%"%tmp_ptr22" = getelementptr i32, i32* %"%tmp_mat_ents", i32 5
store i32 9, i32* %"%tmp_ptr22"
%malloccall.23 = tail call i8* @malloc(i32 ptrtoint (<{ i32, i32, i32* }>* getelementptr (<
%"%tmp_mat_struct" = bitcast i8* %malloccall.23 to <{ i32, i32, i32* }>*
%"%tmp24" = getelementptr inbounds <{ i32, i32, i32* }>, <{ i32, i32, i32* }>* %"%tmp_mat_s
store i32 2, i32* %"%tmp24"
%"%tmp25" = getelementptr inbounds <{ i32, i32, i32* }>, <{ i32, i32, i32* }>* %"%tmp_mat_s
store i32 3, i32* %"%tmp25"
%"%tmp26" = getelementptr inbounds <{ i32, i32, i32* }>, <{ i32, i32, i32* }>* %"%tmp_mat_s
store i32* %"%tmp_mat_ents", i32** %"%tmp26"
%bar = load <{ i32, i32, i32* }>, <{ i32, i32, i32* }>* %"%tmp_mat_struct"
%bar27 = alloca <{ i32, i32, i32* }>
store <{ i32, i32, i32* }> %bar, <{ i32, i32, i32* }>* %bar27
%"%tmp28" = getelementptr inbounds <{ i32, i32, i32* }>, <{ i32, i32, i32* }>* %bar27, i32
%"%tmp29" = load i32, i32* %"%tmp28"
%"%tmp30" = mul i32 %"%tmp29", 1
%"%tmp31" = add i32 %"%tmp30", 2
%"%tmp32" = getelementptr inbounds <{ i32, i32, i32* }>, <{ i32, i32, i32* }>* %bar27, i32
%"%tmp33" = load i32*, i32** %"%tmp32"
%"%tmp34" = getelementptr i32, i32* %"%tmp33", i32 %"%tmp31"
%foo = load i32, i32* %"%tmp34"
%foo36 = alloca i32
store i32 %foo, i32* %foo36
%"%tmp37" = load i32, i32* %foo36
%print38 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @fmt
%baz = alloca i1
store i1 true, i1* %baz
```

```
  %"%tmp39" = getelementptr inbounds <{ i32, i32* }>, <{ i32, i32* }>* %y8, i32 0, i32 1
  %"%tmp40" = load i32*, i32** %"%tmp39"
  %"%tmp41" = getelementptr i32, i32* %"%tmp40", i32 1
  %"%tmp42" = load i32, i32* %"%tmp41"
  ret i32 %"%tmp42"
}

declare noalias i8* @malloc(i32)

define i32 @main() {
entry:
  %n = call i32 @f()
  %n1 = alloca i32
  store i32 %n, i32* %n1
  ret i32 0
}
```

3. Nested function declarations

   Twister code:

```
int a= print("hello");
int d = 4;
fun mf = (x : int) -> int {
int b = 3;
fun sf = ( ) -> int {
fun tf = () -> int
{ int inner = 2; return 4;};
return 2 ; } ;
b = 3 + 4;
List<int> l = { 1 , 2 , 4 } ;
b = d;
int c = 5;
int z = sf();


return b ; } ;


int q = mf(1);
float ax = tofl(3);
```

   LLVM code:

```
; ModuleID = 'Twister'

@"%tmp" = private unnamed_addr constant [6 x i8] c"hello\00"
@fmt = private unnamed_addr constant [3 x i8] c"%s\00"

declare i32 @printf(i8*, ...)

declare i32 @printbig(i32)

declare i8* @fopen(i8*, i8*)

declare i32 @fread(i8*, i32, i32, i8*)

define i32 @tf(i32 %a, i32 %d, i32 %b) {
entry:
  %a1 = alloca i32
  store i32 %a, i32* %a1
  %d2 = alloca i32
  store i32 %d, i32* %d2
  %b3 = alloca i32
  store i32 %b, i32* %b3
  %inner = alloca i32
  store i32 2, i32* %inner
  ret i32 4
}

define i32 @sf(i32 %a, i32 %d, i32 %b) {
entry:
  %a1 = alloca i32
  store i32 %a, i32* %a1
  %d2 = alloca i32
  store i32 %d, i32* %d2
  %b3 = alloca i32
  store i32 %b, i32* %b3
  ret i32 2
}

define i32 @mf(i32 %x, i32 %a, i32 %d) {
entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %a2 = alloca i32
  store i32 %a, i32* %a2
  %d3 = alloca i32
  store i32 %d, i32* %d3
  %b = alloca i32
  store i32 3, i32* %b
  store i32 7, i32* %b
  %malloccall = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* nu
  %"%tmp_list_ents" = bitcast i8* %malloccall to i32*
  %"%tmp_ptr" = getelementptr i32, i32* %"%tmp_list_ents", i32 0
  store i32 1, i32* %"%tmp_ptr"
  %"%tmp_ptr4" = getelementptr i32, i32* %"%tmp_list_ents", i32 1
  store i32 2, i32* %"%tmp_ptr4"
  %"%tmp_ptr5" = getelementptr i32, i32* %"%tmp_list_ents", i32 2
  store i32 4, i32* %"%tmp_ptr5"
  %malloccall.6 = tail call i8* @malloc(i32 ptrtoint (<{ i32, i32* }>* getelementptr (<{ i32,
  %"%tmp_list_struct" = bitcast i8* %malloccall.6 to <{ i32, i32* }>*
  %"%tmp" = getelementptr inbounds <{ i32, i32* }>, <{ i32, i32* }>* %"%tmp_list_struct", i32
  store i32 3, i32* %"%tmp"
  %"%tmp7" = getelementptr inbounds <{ i32, i32* }>, <{ i32, i32* }>* %"%tmp_list_struct", i3
  store i32* %"%tmp_list_ents", i32** %"%tmp7"
  %l = load <{ i32, i32* }>, <{ i32, i32* }>* %"%tmp_list_struct"
  %l9 = alloca <{ i32, i32* }>
  store <{ i32, i32* }> %l, <{ i32, i32* }>* %l9
  %"%tmp10" = load i32, i32* %d3
  store i32 %"%tmp10", i32* %b
  %c = alloca i32
  store i32 5, i32* %c
  %b11 = load i32, i32* %b
  %d12 = load i32, i32* %d3
  %a13 = load i32, i32* %a2
  %z = call i32 @sf(i32 %a13, i32 %d12, i32 %b11)
  %z14 = alloca i32
  store i32 %z, i32* %z14
  %"%tmp15" = load i32, i32* %b
  ret i32 %"%tmp15"
}

declare noalias i8* @malloc(i32)

define i32 @main() {
entry:
  %print = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @fmt,
  %a = alloca i32
  store i32 1, i32* %a
  %d = alloca i32
  store i32 4, i32* %d
  %a1 = load i32, i32* %a
  %d2 = load i32, i32* %d
  %q = call i32 @mf(i32 1, i32 %d2, i32 %a1)
  %q3 = alloca i32
  store i32 %q, i32* %q3
  %ax = alloca double
  store double 3.000000e+00, double* %ax
  ret i32 0
}
```

4. An example of an invalid Twister program and the resulting error printed by our semantic checker: **Twister**

```
fun fill = (r: int, c: int, x: int) -> Matrix<int> {
    Matrix<int> M = Matrix(r,c);
    if (x < 2) {
        for (i in range(0,r)) {
            for (j in range(0,c)) {
            }
        }
   } else {
        return 2;
    }
};
```

Semant error

```
Error: "
Error: "
Error: "
Statement ’’
return 2;
’’
 would return invalid type int where return values should be of type Matrix<int>
" @ statement: return 2;
" @ statement: if (x < 2) {
for i in range(0,r): {
 for j in range(0,r): {
 M[i][j] = x;
}
}
return M;
} else {
return 2;
}
```

```
" @ statement: fun fill = (r : int,c : int,x : int) -> Matrix<int> {Matrix<int> M = Matrix(r,
if (x < 2) {
for i in range(0,r): {
 for j in range(0,r): {
 M[i][j] = x;
}
}
return M;
} else {
return 2;
}};
```

The number of **Error: "** lines indicates how many stack levels deep the error is; each closing quote indicates the end of one stack level. The innermost stack level, which is the first one printed, is where the error occurred; the outer stack levels are reported to help identify the location of the error in case the same variable names or statements occur in multiple places in a large file. Having all the stack levels is usually sufficient information because at the outermost level only one function in the entire file can be declared with a specific name in the outermost scope; only if lots of identical if/else statements or for loops appear would the error be hard to identify, but it's bad coding style anyway to have lots of identical looking code in the same program.

8 Lessons Learned

Tian: I have learned a lot from this project. My teammates were resourceful and helpful. Anand helped me whenever I have problem writing the tests. I want to thank him for that.

Anand: I learned that I need to exercise more self-restraint when I am tempted to propose ambitious goals, because attempting to implement too many difficult features can be a waste of time and cause a worse final product when scope is cut at the last minute to focus on the essentials, and also that I need to be more assertive when asking people to complete things on time or work harder to complete things when deadlines have already been missed.

Arushi: I developed an appreciation for the level of detail that goes into designing a language. As someone who used computer languages but never really studied them or analyzed them in depth, this made me realize that the functionality and lack of ambiguity of the programming languages I use requires a great deal of careful and detailed thought.

9 Appendix