Columbia University
COMS W4115: Programming Languages and Translators

Opposing Discrete and Definite Heuristics: A simple mathematical distribution language

FINAL REPORT

Alex Kalicki (avk2116)
Alexandra Medway (afm2134)
Daniel Echikson (dje2125)
Lilly Wang (lfw2114)

Contents

1. **Introduction**
   - 1.1. Philosophy and Motivation
   - 1.2. Language Description
   - 1.3. Simple Example
   - 1.4. Installing and Running *Odds*

2. **Lexical Conventions**
   - 2.1. Identifiers
   - 2.2. Reserved Words
   - 2.3. Literals
     - 2.3.1. Numerics
     - 2.3.2. Boolean
     - 2.3.3. String
   - 2.4. Punctuators
   - 2.5. Comments

3. **Types, Operators, and Expressions**
   - 3.1. Basic Types
   - 3.2. Lists
   - 3.3. Arithmetic Operators
   - 3.4. Relational Operators
   - 3.5. Equality Operators
   - 3.6. Logical Operators
   - 3.7. Binding Operators and Expressions
     - 3.7.1. Binding using `=`
     - 3.7.2. `do`
   - 3.8. Precedence and Order of Evaluation

4. **Control Flow**
   - 4.1. Statements
   - 4.2. Identifier Scope
   - 4.3. `If, Then, Else`

5. **Functions and Program Structure**
   - 5.1. Function Definition
   - 5.2. Calling Functions
   - 5.3. Nested Functions
   - 5.4. Anonymous Functions
   - 5.5. Caked Function Calls
   - 5.6. Recursion

6. **Distributions**
   - 6.1. Definition

6.2. Declaration
    6.2.1. Continuous Distribution
    6.2.2. Discrete Distribution
6.3. Built-in Distributions
6.4. Operations
    6.4.1. Sampling
    6.4.2. Addition and Subtraction
        6.4.2.1. Constants
        6.4.2.2. Distributions
    6.4.3. Multiplication and Division
        6.4.3.1. Constants
        6.4.3.2. Distributions
    6.4.4. Exponentiation
6.5. Usage
    6.5.1. Function Application
    6.5.2. Simulation

7. Standard Library
7.1. List Operations
7.2. Distribution Operations
7.3. Mathematical Constants
7.4. Printing

8. Project Plan
8.1. Process
    8.1.1. Planning
    8.1.2. Specification
    8.1.3. Development
    8.1.4. Testing
8.2. Style Guide
8.3. Timeline
8.4. Roles and Responsibilities
8.5. Software Development Environment

9. Architectural Design
9.1. Compiler Overview
9.2. Scanner
9.3. Parser and AST
9.4. Analyzer and SAST
9.5. Pythonizer and PAST
9.6. Generator

10. Test Plan
10.1. Source to target examples
10.2. Test Suites

10.2.1. Scanner
10.2.2. Parser
10.2.3. Compiler
10.2.4. Standard Library Functions
10.3. Test Automation

11. Lessons Learned
11.1. Alex Kalicki
11.2. Alexandra Medway
11.3. Danny Echikson
11.4. Lilly Wang

12. Appendix
12.1. Compiler Code
12.2. Test Code
12.2.1. Scanner
12.2.2. Parser
12.2.3. Compiler
12.2.4. Standard Library
12.3. Git Log

1. Introduction

1.1 Philosophy and Motivation

I see your boundless form everywhere, the countless arms, bellies, mouths, and eyes; Lord of All, I see no end, or middle or beginning to your totality" - Arjuna to Krishna, Bhagavad-Gita.

As programmers, we are often forced to think and program in terms of definite binaries: 0 or 1, if-else, do-while, one answer or some finite number of answers. The real world, however, is not so determinate or discrete. The real world is fluid. The real world operates on chance and spectrums of possibility. As Arjuna remarks, the problems we seek solutions to frequently have no apparent beginning, middle, or end. They must be conceived of in their totality. We understand this to be the programmer's job.

The programmer must take real-world problems - problems that present themselves as neither obviously discrete nor definite - and come up with solutions that can be computed on machines that operate within the realm of the discrete and definite. We understand the programmer to be a translator of sorts, from the uncertainty of the real to the general certainty of the virtual. The motivation for *Odds* is to ease this process of translation. We recognize the need to be able to compute not only on definite values, but also on discrete and non-discrete distributions, and continuous ranges of numbers. In implementing these structures as an essential part of *Odds*, we hope to create a programming language that more seamlessly reflects the manner in which problems and solutions are posed in the real world, that is, the world of fluidity and uncertainty.

1.2 Language Description

Odds is a functional programming language that uses a simple and straightforward syntax. *Odds* centers around mathematical distributions and expresses operations on them in a direct and uncomplicated way.

Distributions support standard operations such as addition and multiplication. In addition to these simple operations, users have the option of sampling the distribution in order to apply complex calculations on portions of the data.

1.3 Simple Example

Here is a very simple example of the classic "Hello World" in *Odds*:

```
do print(“Hello World”)
```

It is also not difficult at all to create a `dist` type and perform different operations on it, which is at the heart of our language. In this example, we create a distribution that has a range of -3 and 3. It applies the normal probability density function over this range.

```
do d = <-3, 3> | normal |
```

1.4 Installing and Running *Odds*

To install *Odds*, clone the GitHub repository found at: https://github.com/odds-lang/odds. (requires Python 2.7 and Ocaml 4.02). To install necessary dependencies, run

```
pip install -r config/requirements.txt
```

Inside the odds directory, type `make`. This will generate all the files necessary to translate an *Odds* file to its translated Python code. Write a simple *Odds* file with a file format of `.ods`. Let’s say for this example, we have a file entitled “`hello_world.ods`” with the code for “hello world” as written above.

The entry point to the compiler is through odds.sh. The following format is how the script should be run:

```
odds.sh <flag> [input_ods_file] [output_python_file]
      -c    Compile odds input_file to python code in output_file with stdlib
      -r    Compile odds input_file into raw python output_file (without stdlib)
      -s    Print odds input_file as semantically checked ast
      -h    Display this list of options
```

For most purposes, users should compile with the `-c` flag in order to get an executable Python translated file. The script takes the *Odds* file and an output python file (it does not have to be an existing file). To get “`hello_world.py`”, just run:

```
odds.sh -c hello_world.ods hello_world.py
```

2. Lexical Conventions

2.1 Identifiers

Identifiers in *Odds* consist of a combination of alphabetical characters, underscores, and numbers. Numbers and underscores are forbidden to be the first character of identifiers. Identifiers are case sensitive.

Valid	Invalid

`helloWorld`
`h3ll0W0r1d`
`hello_world`
	`3ll0W0r1d`
	`hello.World`
	`hello World`
	`_hello_world`

2.2 Reserved Words

Odds reserves the following words. They may not be used in a program as identifiers:

```
do  if  then  else  return  true  false  void
```

2.3 Literals

2.3.1 Numerics

In *Odds*, there are five types of literals. The three primary ones are Num, String, and Bool. The final literal types, void and list, will be discussed in sections 3.1 and 3.2.

There is only one numeric literal type - Num. Num literals consist of either:

a) a sequence of digits with an optional negative sign in front to indicate negativity,

or

b) optional digits in the beginning followed by a period and at least one digit after the period, with an optional negative sign in front to indicate negativity.

Under the hood, Odds treats integers and floating point numerals the same and they can be used interchangeably in Odds code. Floating point number comparisons are valid, but strict comparison might not be advisable for full precision. Operations between integers and floating point numerals return a floating point result. The following table depicts valid integer and floating point literals.

Valid	Invalid
6 -192 04 1.67 .4 -234.19	1,000 +15 1. 4.1.1.5 +6.23

2.3.2 Boolean

Odds also has boolean literals that can be one of two values: `true` or `false`. They denote the values of logical true and logical false respectively.

2.3.3 String

String literals are delimited by double-quotes. To place a double-quote within a string, a backslash is placed before the double-quote to escape it.

2.4 Punctuators

Type	Explanation
( )	Function declarations, Function Calls
< >	Distribution type, operators
[ ]	Lists
,	List Delimiter, Distribution Range Delimiter
| |	Distribution function Delimiter
->	Function Delimiter
“	String Delimiter

2.5 Comments

Odds supports multi-line comments. A backslash followed by an asterisk denotes the start of a comment, and an asterisk followed by a backslash denotes the end of the comment. Comments may not be nested and must be closed before the end of the file is reached.

	Comment

Valid	`/* I can write anything here */`
	
	`/*`
	`* This is a`
	`* multiline`
	`* comment`
	`*/`

Invalid	`/* Nested comments are /* not */ okay */`

3. Types, Operators, and Expressions

3.1 Basic Types

Odds has five basic types. These basic types are explained below:

Type	Example	Explanation
Num	`1.24, .23, -27.0, 1, -213, 24`	The Num type might be either a Double-precision 64-bit IEEE 754 floating point number or a signed 32-bit two's complement integer.
Bool	`true, false`	The Bool type has only two possibilities: `true` or `false`. Used in control flow.
String	`"23%", "Edwards is a great and benevolent teacher.", "y&632@", "say: \" hello, world\""`	The String type is a sequence of characters. Note *Odds* has no char type, thus even single characters are expressed as strings. All strings are delimited by double-quotes. To place a double-quote within the string itself, escape with backslash.
Void	`void`	The Void type has only one value, void. It is used to represent the return type of expressions that return 'no value.' Expressions that are evaluated only for their side-effects return the void type.

3.2 Lists

Lists are *Odds'* basic ordered collection type. They are homogenous, i.e. consisting of only one data type. They are singly linked lists and thus have *O(1)* insertion time but *O(n)* access time. Lists are non-mutable.

Lists are delimited by square-brackets and the values within the list are comma-separated. They may be initialized as empty or containing any number of values. Below are a few examples of binding a list to an identifier using list literals:

```
do one_to_six = [1, 2, 3, 4, 5, 6]
/*one_to_six is a list of Nums from 1 to 6 */

do Bool_list = [true, false, false]
/*Bool_list is a list of Bools*/

do empty  = []
/*empty is an empty list*/

do error_list = [42, “STAR WARS IS BETTER THAN STAR TREK!”]
/* error_list is an illegal list and
 * will throw an error because it is
 * non-homogenous. It has a Num and a String.
 */
```

3.3 Arithmetic Operations

Odds' has six basic arithmetic operators. All arithmetic is floating point arithmetic.

Operator	Example	Explanation
**	2 ** 3.0, 3.24 ** -3.0	The Exponentiation operator takes the number on the left-hand side and raises it to the power of the number on the right-hand side.
*	2.72 * 4, -2 * 4	The Multiplication operator takes the number on the left-hand side and multiplies it by the number on the right-hand side.
/	-3 / 3, 27.2 / .2	The Division operator takes the number on the left-hand side and divides it by the number on the right-hand side.

`+`	`3 + 3,` `4.72 + -.72`	The Addition operator takes the number on the left-hand side and adds it to the number on the right-hand side.
`-`	`8.34 - 9.37,` `8 - -2`	The Subtraction operator takes the number on the right-hand side and subtracts it from the number on the left-hand side.
`%`	`6 % 9,` `7.2 % -3.2`	The Modulo operator takes the number on the left-hand side and mods it by the number on the right-hand side.

3.4 Relational Operators

Odds has four basic relational operators. All return a Bool: `true` or `false`.

Operator	Example	Explanation
`<`	`1 < 2.54,` `2.0 < -74.2`	The Less-Than operator tests if the number on the left-hand side is less than the number on the right-hand side.
`>`	`4 > 7,` `.2 > 42.2`	The Greater-Than operator tests if the number on the left-hand side is greater than the number on the right-hand side.
`<=`	`2.0 <= .2`	The Less-Than-Or-Equal operator tests if the number on the left-hand side is less than or equal to the number on the right-hand side.
`>=`	`2 >= 42,` `-7.0 >= 1.4`	The Greater-Than-Or-Equal operator tests if the number on the left-hand side is greater than or equal to the number on the right-hand side.

*Note that not all the examples above evaluate to `true`.

3.5 Equality Operators

Odds has two equality operators == and `!=`. All equality operations return a `Bool`. The behavior of == and `!=` with various types is described below.

Types	Example	Explanation
`Num == Num`	`1.4 == 2 /* false */`	True if both nums are the same
`Num == Bool`	`1 == true /* true */` `2 == true /* false */`	1 evaluates to `true`, all other nums evaluate to false
`Num == non-Num/non-Bool`	`1 == "hi" /* false */`	Always evaluates to `false`
`String == String`	`"hi" == "hi" /* true */` `"hi" == "ih" /* false */`	Structural comparison of `Strings`
`String == non-String`	`"hi" == true /* false */` `"hi" == false /* false */`	Always evaluates to `false`
`List == List`	`[1,2] == [1, 2]/* true */` `[1] == [2]  /* false */`	Structural comparison of two Lists
`List == non-List`	`[1,2] == "easter egg"` `/* false */`	Always evaluates to `false`
`Dist == Dist`	`<0, 1>|normal| ==` `<0,1>|normal| /* false */`	Always evaluates to `false`
`Dist == non-Dist`	`<0, 1>|normal| == 1`	Always evaluates to `false`
`Void == Void`	`void == void  /* true */`	`True`
`Void == non-Void`	`void == 0 /* false */` `void == false /* false */`	Always `false`

`Expression1 != expression2` is equivalent to ! `expression1 == expression2`.

3.6 Logical operators

Odds has three logical operators. The expressions on each side of the operator must evaluate to a Bool. All operators return a Bool: `true` or `false`.

Operator	Example	Explanation
`&&`	`true && false`	Logical And
`||`	`true || false`	Logical Or
`!`	`! true`	Logical Not

3.7 Binding Operators and Expressions

Remember, all expressions in *Odds* return a value.

3.7.1 Binding using =

To bind an identifier to a value or function, one uses the = operator.

All bindings follow the pattern *x* = *y* where *x* is the identifier and *y* is the value or function that the identifier, *x*, is being bound to.

For example:

```
do num = 7
/* binds the integer 7 to the identifier num */

do A_pls = “Edwards is a techer”
/* binds a string literal to the identifier A_pls */

do is_true = true && (true || false)
/* binds the result of a logical expression to the identifier is_true */
```

Additionally, assignments may be chained. For example:

```
do l1 = l2 = [1,2,3]
/* l1 and l2 are both lists that contain the nums 1, 2, and 3 */
```

3.7.2 do

Though all expressions return a value in *Odds*, sometimes there is no need to capture the actual return value. Using do allows you to ignore the return value.

For example,

```
do print(3.14159265)
/* calls the function, print, which
 * prints the number “3.14159265”. Then
 * ignores the value print returns.
 */
```

3.8 Precedence and Order of Operations

The precedence of operators is listed below from highest precedence to lowest precedence:

Operators	Explanation
`**`	Exponentiation
`|**`	Distribution-Num Exponentiation
`*, /, %, <*>`	Multiplication, Division, Remainder, Distribution Multiplication
`+, -, <+>`	Addition, Subtraction, Distribution Addition
`|*, |+`	Distribution Stretch, Distribution Shift
`<=, >=, <, >, ==, !=, <>`	Relational Operators, Equality Operators, Sample
`!`	Logical NOT
`&&`	Logical AND
`||`	Logical OR
`::`	Cons
`IF-THEN-ELSE`	Control-flow
`=`	Assignment

A subset of the operators above will be discussed further in later sections.

4. Control Flow

The control-flow statements of *Odds* specify the order in which computation is to be performed, as well as decision-making about which computations should be run.

4.1 Statements

An expression such as `x = 0` or `i + 1` or `print(...)` becomes a *statement* when it is preceded by the keyword `do`, as in

```
do x = 0
do i = i + 1
do print(“Go team!”)
```

4.2 Identifier Scope

Identifiers in *Odds* are scoped to the function in which they are enclosed. There is no concept of global variables in the language. An identifier declared in a function is local to that function - `x` in function `func1` is independent from `x` declared in function `func2`. After returning from a function, any attempt to reference an identifier declared within that function results in undefined behavior as the identifier is considered out of scope and no longer usable.

```
do x = 4
do increment = (n) ->
       do x = 1            /* does not affect outer x */
       return n + x
do y = increment(x)        /* y == 5 */
/* x is still equal to 4 */
```

Identifiers referenced within functions that have not been defined in the function take on their previous value in the program. If the identifier has not been defined previously, an error results:

```
do a = 5
do adda x = x + a               /* x + 5 */
do a = 10
do print(adda(0))               /* 5 */
do adda x = x + a               /* x + 10 */
do print(adda(0))               /* 10 */
```

The following section will discuss defining and calling functions in further detail.

4.3 If, Then, Else

The if-then-else structure is used to express decisions and program control based off the results of those decisions. Formally, the syntax is

```
if expression1 then expression2 else expression3
```

expression1 is evaluated; if it is equivalent to the boolean value `true`, *statement2* is executed. Conversely, if *expression1* has the boolean value `false`, *statement3* is executed instead.

Because of the mandatory if-then-else structure, conditionals can be nested:

```
if expression1 then
       if expression2 then expression3 else expression4
else expression5
```

The above code will check *expression1*; if *expression1* evaluates to `true`, then the program will proceed to check *expression2*, evaluating *expression3* on `true` and *expression4* on `false`. If *expression1* evaluates to `false`, the program will instead evaluate *expression5*.

The mandatory structure also makes writing unambiguous multi-way else-if conditionals a breeze:

```
if expression1 then expression2
       else if expression2 then expression3
       else if expression4 then expression5
       else expression6
```

Since the body of a conditional statement only allows a single expression, if the user wants to put multiple statements in the body of a conditional, he or she puts them in an anonymous function and immediately invokes it. For example:

```
if expression1 then
       (() ->
              do expression2
              do expression3
              /* . . . */
              return expression 4
       )() /* create anonymous function and immediately invoke it */
else expression5
```

5. Functions and Program Structure

Functions allow programmers to break large tasks into smaller ones in order to allow for better code reuse and more readable work. *Odds* makes it easy to define and call your own functions in order to write algorithms and perform complex tasks.

5.1 Function Definition

In order to be called in various parts of a user program, a function must first be defined. Functions definitions take the form:

```
(argument_1, argument_2, …, argument_n) ->
       declarations and statements
       return statement
```

In order to better understand the process of function definition in *Odds*, let us break this function into its respective components and examine each individually. To begin with, we have the function signature:

```
(argument_1, argument_2, …, argument_n)
```

The function signature defines a list of identifiers used to refer to user-passed arguments within the function. The function name may be followed by a list of one or more argument names, each of which corresponds to a mandatory value that must be passed when the function is called. All arguments in *Odds* are passed by value.

The function signature is followed by the delimiter "`->`", and an arbitrarily long list of declarations and statements. These statements can declare local function identifiers, call other functions, or execute complex control flow logic as described in the preceding sections.

Finally, the function must end with a mandatory `return` statement. This statement uses the `return` keyword, followed by the value the function will return when called. Functions in which it is not necessary to return a value must end with `return void`.

Function definitions are usually bound to identifiers using the `=` operator, just as we would normally bind values to identifiers. In the previously given example,

```
do increment = (n) ->
       do x = 1
       return n + x
```

we are defining a function that takes one argument, bound to `n` within the function, and returns `n+1`. We bind this function to the identifier `increment` for later use. The code

```
do is_sum_even = (a, b) ->
       return if a + b % 2 == 0 then true else false
```

defines a function that takes two arguments with identifiers `a` and `b` within the function, and returns the boolean `true` if the sum of the arguments is even or `false` if the sum is odd. The code binds this function to the `is_sum_even` identifier using the `do` statement.

5.2 Calling Functions

Calling previously defined functions in *Odds* is extremely straightforward and mimics the workflow found in many other contemporary languages. Given a function

```
do increment = (n) -> return n + 1
```

you call the function by listing its identifier followed by the arguments to be passed in:

```
do print(increment(4))  /* 5 */
```

Anonymous functions, described below, are called by defining the function and immediately providing arguments with which to call the function.

5.3 Nested Functions

Odds supports the definition of nested functions in order to further break down functionality into its component parts. While these types of functions are most useful when making recursive calls, as discussed below, they can also be used for more straightforward computation as well.

Nested functions are defined within their enclosing functions just as local, non-function identifiers are bound. The nested function, being local to its enclosing shell, can not be called outside the scope with which it is defined:

```
do is_sum_even = (a, b) ->
       do is_even = (n) -> return n % 2 == 0
       return is_even(a + b)
do is_sum_even(2, 4)             /* true */
do is_even(5)                    /* error */
```

5.4 Anonymous Functions

In addition to declaring functions and binding their definition to identifiers, users can also define “anonymous functions” that are not bound to a name. These functions are often

applicable when the functionality is only needed for an ephemeral amount of time to render a direct result:

```
do is_4_even = ((n) -> return n % 2 == 0)(4)
/* is_4_even == true */
```

Anonymous functions can also be used as a return type. Here, the function normal takes a mean and standard deviation and returns a function mapping a value `x` to its weight within the distribution:

```
do my_normal = (mean, stdev) ->
       return (x) ->
              do exp = -1.0*((x - mean)**2.0 /(2.0 * stdev)**2.0)
              return 1 / (stdev * (2.0 * PI) ** (0.5)) * EUL ** exp

do my_standard_normal = normal(0.0, 1.0)
do print(my_standard_normal(0.5))
```

5.5 Caked Function Calls

A caked function call is the creation of an anonymous function and then its immediate invocation. The syntax for caked function call is just like a regular function call - `call(params)` - except instead of a function identifier followed by parentheses and arguments, the user defines the function and then after the definition calls it with arguments and parentheses - *`(function)(params)`*. For example:

```
do four = 2 + ((x, y) -> return x + y)(1, 1)
```

5.6 Recursion

Like many languages, *Odds* supports the ability for a function to recursively call itself. Such functions typically have a base case that defines the point at which recursion ends and a recursive call if it has not yet ended. For example, one could define the Euclidean algorithm as follows:

```
do gcd = (a, b) ->
       return if b == 0 then a else gcd(b, a % b)
do print(gcd(48, 36))  /* 12 */
```

6. Distributions

6.1 Definition

There are two types of distributions in *Odds*, continuous distributions, and discrete distributions. A continuous distribution is a *domain* (measurable set of data) to which a function of a discrete variable is applied. This function will map the set of data to a new set of weighted outcomes. A discrete distribution is composed of a distinct set of values, and is weighted by the corresponding probabilities of these random values occurring.

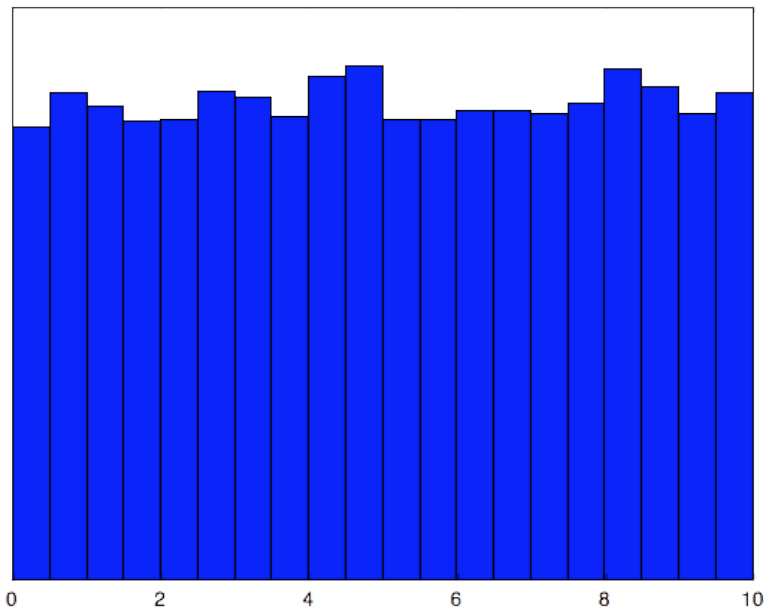
6.2 Declaration

6.2.1 Continuous Distribution

To create a continuous distribution in *Odds*, users need to specify the measurable set they are interested in. The measurable set of values, also called a *range*, indicates a continuous or discrete set of numbers. Declaring a distribution over the range 0 to 10 can be done in the following way:

```
do a = <0, 10> | (n) -> return 1 |
```

The distribution above, `a`, is a uniform distribution and can be visualized with the following graph:



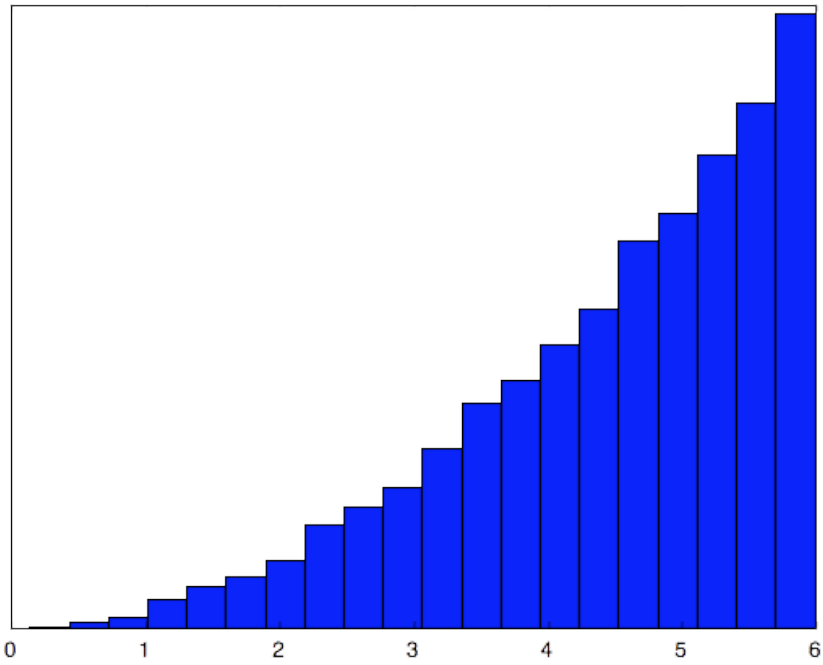
Let's create a new distribution which has a more complicated function associated with it. Functions associated with distributions, also called *maps*, are a bit different from standard functions within *Odds*. *Maps* take in one parameter of type Num, and return one parameter of type Num. Let's create a function with these specifications called 'squared.' This function will map the input x to an output, x^2. We declare our function in the following way:

```
do squared = (x) -> return x**2
```

We can then create our distribution d in the following way:

```
do b = <0, 6> | squared |
```

This creates a distribution, `b`, which can be visualized with the following graph:



Applying a function to a distribution creates weight within the distribution. Looking at the graph above, we see that a value of 4 with a weight of 16 is 4 times more likely to occur than a value of 2 with a weight of 4. Compare this to `a` in which each value is equally weighted: a value of 4 is equally as likely to occur as a value of 2.

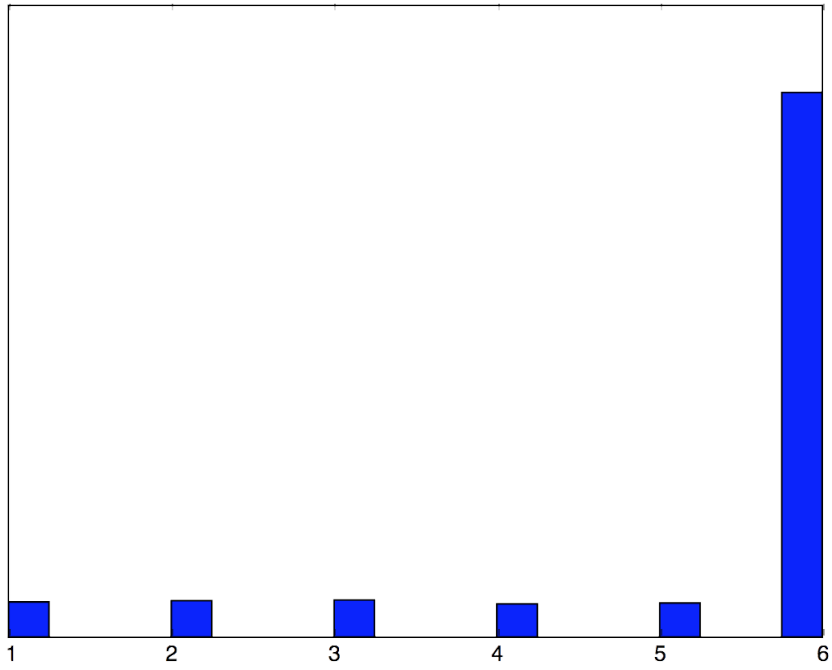
6.2.2 Discrete Distribution

To create a discrete distribution in *Odds*, users need to specify both the values and the corresponding outcome probabilities for those values. The values and weights should

be passed into the distribution in the form of two lists, where the index of the value corresponds to the index of the that value's probability of occurring. Imagine we have a weighted die, and rolling the die will result in a six 75% of the time, and each of the other numbers 5% of the time. Declaring a distribution which represents the outcomes of this weighted die can be done in the following way, where each value corresponds to the rolled outcome:

```
do die = |<[1,2,3,4,5,6], [5,5,5,5,5,75]>|
```

The distribution `die` can be visualized with the following graph:

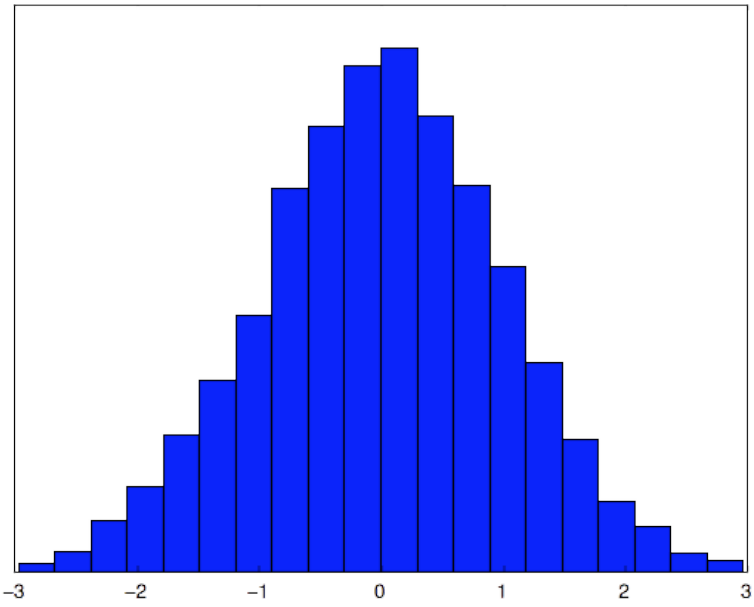


6.3 Built-In Distribution Functions

Distributions can be used for a number of purposes, including statistical probability. *Odds* has two probability distributions already included in its standard library: the *uniform* distribution and the *normal* distribution. To apply these distributions to a range of numbers, the user can declare a distribution in the following way using the built-in probability distribution keyword:

```
do c = <-3, 3> | normal |
```

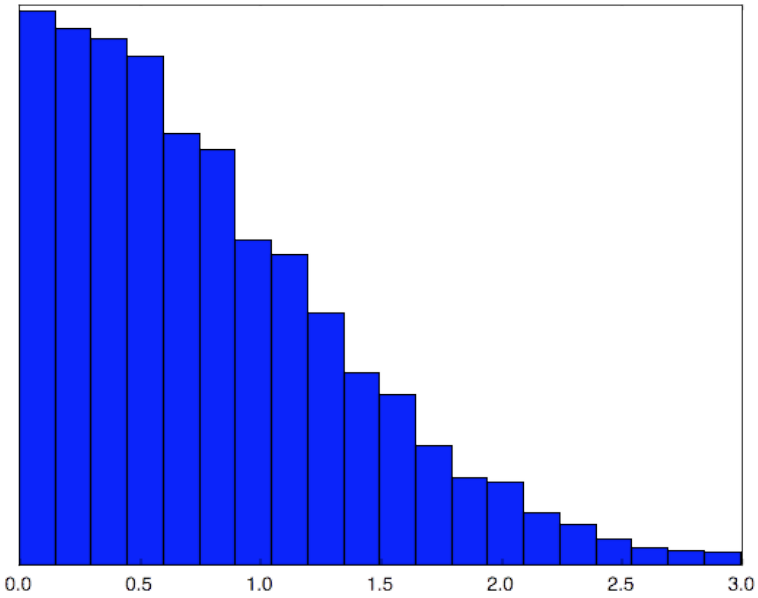
This creates a distribution, `c`, over the range -3 to 3 with the *normal* distribution applied.



The normal distribution is centered about 0 with a standard deviation of 1. If the user doesn't center the range about 0, the distribution will be skewed.

```
do x = <0, 3> | normal |
```

In a visual representation of `x`, we can see that the data is skewed.



To create a normal distribution centered around a point other than 0, with a standard deviation other than 1, users must use *operators* as described in the next section.

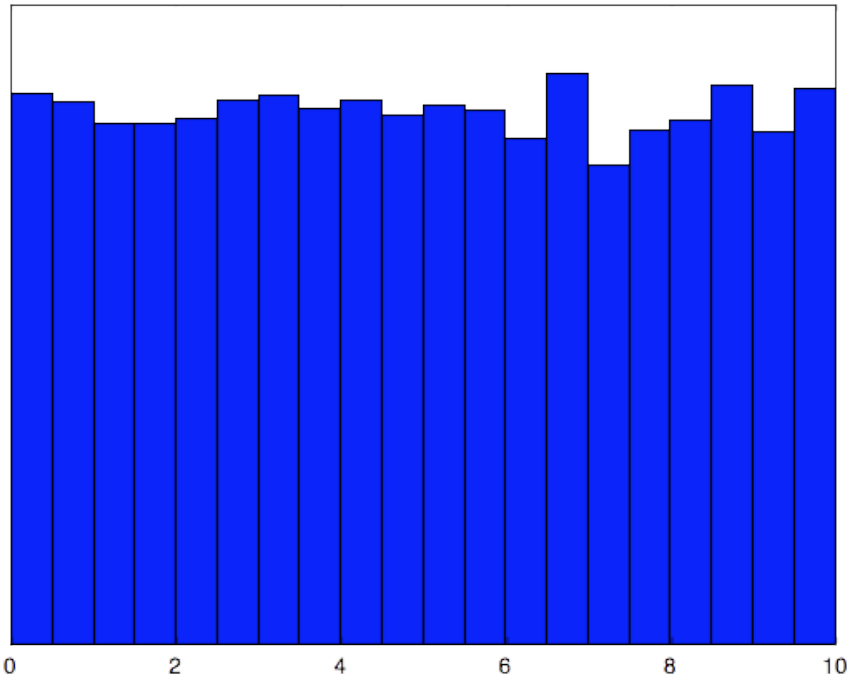
6.4 Operations

6.4.1 Sampling

There are a number of operations one can use on distributions, the first of which is *sampling*. Sampling has several advantages, as it allows the user to work with discrete values rather than a continuous range. The sample operator applies a number to a distribution, and returns a list whose length is the same as the number. Please see the standard library section for ways to manipulate and use a list. The sampling operator can be used on a distribution, d, in the following way:

```
do d = a <> 100
```

We had defined `a` to be a uniform distribution across the range 0 to 10. Thus, we would expect our sample of 100 values to reflect the continuous distribution.



As one can see in the example above, the 100 sampled values reflect the overall distribution very well.

Sampling is especially useful with statistical distributions. We create a second sample, `sample`, below using `d` from above.

```
do sample = d <> 10
```

We defined d above to be a normal distribution over the range -3 to 3. Looking at Figure C one sees the percentage of values distributed within a specific standard deviation range. In the normal distribution, 68% of the values are within one standard deviation from the mean. We would expect our sample to also have ~68% of its values between -1 and 1. In `sample` above, *Odds* would select the 10 values randomly from within the `d`, given the weight of its distribution.

6.4.2 Addition and Subtraction

6.4.2.1 Constants

To transform a distribution by shifting it a constant value, one has the option of adding or subtracting constants to it.

```
do distribution = <-1, 1> | normal |
/* Normal distribution from -1 to 1 */

do transformation = distribution |+ 5
/* Transformation is now a normal distribution
 * ranging in values from 4 to 6, with a mean
 * of 5, and the same standard deviation as
 * the original distribution
 */
```

6.4.2.2 Distributions

Two distributions of different variables can be combined into a single distribution of one variable by using the addition or subtraction operator.

```
do e = <0, 3> | uniform |  /* Uniform distribution */
do f = <3, 6> | uniform |  /* Uniform distribution */
```

It helps to think about a distribution as a sampled list of discrete values rather than continuous values when visualizing addition. Let's choose two samples, one from `e` and one from `f`, with precision value of 1. Thus we are working with two lists of `[0, 1, 2, 3]` and `[3, 4, 5, 6]`. Adding the values of a distribution is not as simple as adding the lists together. Rather, each element in the first distribution must be summed with each element in the second distribution. We can then visualize addition with the following:

$$\sum\sum (e[i] + f[j])$$

As one can see, the smaller the step value between each i and j, the more precise the addition. Ideally, the step value is 0, and the distribution is entirely continuous rather than discrete. Rather than performing this complicated mathematical operation, *Odds* makes addition as simple as the following:

```
do g = e <+> f
```

The calculation of `g`, with precision value 1, would result in the following list:

```
[3, 4, 5, 6, 4, 5, 6, 7, 5, 6, 7, 8, 6, 7, 8, 9]
```

Sorted, we get the following:

```
[3, 4, 4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 8, 8, 9]
```

Notice that we no longer are working with a uniform distribution, as the value 6 appears 4 times, whereas the value 3 appears only once. Thus, 6 has a heavier weight than 3, and when sampled, is 4 times more likely to occur.

However, `g` does not have to be treated like the discrete list of numbers we see above, and can be treated as a continuous distribution which represents the combination of `e` and `f` with the operation from above.

Addition, like all distribution operations, seamlessly works on both continuous and discrete distributions. A user can mix discrete and continuous distributions using these operations.

6.4.3 Multiplication

6.4.3.1 Constants

Multiplying a distribution by a constant greater than one *stretches* the values and their corresponding weights. Multiplying a distribution by a constant less than one *contracts* the values and their corresponding weights.

```
do h = <-1, 1> | normal |
do i = h |* 3
/* i is a normal distribution across the range -3 to 3
 * centered about 0 with a standard deviation of 3
 */
```

6.4.3.2 Distributions

Just like addition, the multiplication operator will combine two distributions into one. If we have `e` and `f` from above, then the multiplication of the two distributions would be done with the following mathematical formula:

$$\sum\sum (e[i] * f[j])$$

To do this calculation in *Odds*, a user can create `j` – the multiplication of `e` and `f` – with the following line of code:

```
do j = e <*> f
```

6.4.4 Exponentiation

Distributions can only be exponentiated by a constant. This applies the exponentiation operator to every value in the distribution, as if the distribution were to be treated as discrete values.

```
do u = <1, 2> | uniform |
/* u is a uniform distribution across the range 1 to 2 */

do v = u |** 2
/* v is a distribution across the range 1 to 4, with u's weights applied to the
stretched elements of v */
```

6.5 Usage

Distributions allow a user to work with a large range of values rather than a small sample of discrete numbers. This poses a number of advantages to working with standard lists.

6.5.1 Function Application

If a user is interested in the way a function will affect a range of data, they can create a distribution and apply functions in order to transform it. This is far simpler than creating a list, and running loops to apply the function, especially if the user is unsure how precise they want their sample to be. To work with the discrete values, the user always has the option of sampling the data.

6.5.2 Simulation

Going back to our weighted dice example from before, we can use distribution addition to simulate "rolling" the die twice. Getting the probability distribution of the "sum" of two

weighted dice rolls can be done by adding the distribution `die` to itself. This creates a new distribution with minimum value 1, and maximum value 12.

```
do two_rolls = die <+> die
```

We could then sample once to obtain a two-die roll following the correct distribution:

```
do roll_value = two_rolls <> 1
```

7. Standard Library

Odds' standard library comes with a number of operations and functions for list and distribution manipulation as well as useful mathematical constants. Theses functions are automatically included when the user compiles a program using *odds*' -c flag.

7.1 List Operations and Standard Library

Operator	Example	Result	Explanation
head	`head([1, 2, 3])`	`1`	Returns the head of the list
tail	`tail([true, true, false])`	`[true, false]`	Returns a list of all elements but the head
`::`	`1 :: [2, 3]`	`[1, 2, 3]`	Append an element to the beginning of the list in *O(1)* time.
len	`len([1,2,3])`	`3`	Returns the length of the list
list_empty	`list_empty([])`	`true`	Returns true if list is empty
list_make	`list_make(3, 0)`	`[0, 0, 0]`	Takes in two arguments, the size of the list to be created, and a value with which to initialize

list_get	`list_get(2, [1, 2, 3])`	`3`	Returns an element at a specific index in a list

list_fold	`do add =`
	`(a, b) ->`
	`return cur + str`
	
	`list_fold(add, 0, [40, 2])`
		`42`	Apply a function to a partial result and an element of the list to produce the next partial result. Moves from the head of the list to the tail

list_rev	`list_rev([1, 2, 3])`	`[3, 2, 1]`	Reverse the order of the elements of a list
list_concat	`list_concat([1,2], [3,4])`	`[1,2,3,4]`	Returns the first list concatenated with the second
list_map	`do plus1 = (x) -> return x + 1` `list_map(plus1, [1, 2, 3])`	`[2, 3, 4]`	Apply a function to each element of a list to produce another list

list_iter	`list_iter(print, [“1”, “2”, “3”])`	`/*prints*/`
		`1`
		`2`
		`3`
			Apply a function to each element of a list; produces a void result.

list_insert	`list_insert(1, 2, [1, 3])`	`[1, 2, 3]`	Inserts the specified value before the specified index
list_remove	`list_remove(0, [0,1,2,3])`	`[1, 2, 3]`	Removes the value at the specified index

7.2 Distribution Operations and Standard Library

Operator	Example	Result	Explanation
uniform	`let u = <0, 1>|uniform|`	`/* u is a uniform distribution from 0 to 1 */`	A uniform distribution map function
normal	`let n = <-1, 1>|normal|`	`/* n is a normal distribution from -1 to 1 */`	A normal distribution map function

E	E(n) /* n is distribution from above */	0.0	Returns the expected value of the distribution
P	P(0, n) /* n is distribution from above */	.5	Return the probability that the distribution is less than the value passed as the first arg
<+>	do b = n <+> n /* n is distribution from above */	/* b is convolution of n with n */	Return a new distribution which represents the addition (convolution) of the two distributions.
<*>	do c = n <*> u /* using distributions n and u from above */	/* c is the production distribution of u and n */	Return a new distribution which represents the multiplication (product distribution) of the inputted distributions.
|+	do d = n |+ 3	/* e is a new distribution which is the same as n, with all elements increased by 3 */	Returns a new distribution which has been shifted by a numeric value.
|*	do e = n |* 3	/* f is a new distribution which has the same shape as n, but with a new standard deviation of 3 */	Returns a new distribution which has been stretched by a numeric value.
|**	do f = n |** 3	/* f is a new distribution which has exponentiated the elements in n by 3 */	Returns a new distribution which has been exponentiated by a numeric value.
<>	do g = f <> 10	/* g is a list with 10 elements which have been randomly sampled from distribution f */	Returns a list, of the sample size specified by the user, which has been randomly sampled from f.

7.3 Mathematical Constants

Because *Odds* can be used for numerous mathematical purposes, such as modeling and distribution, the user is provided with built-in mathematical constants.

Constant	Value	Definition
`PI`	`3.14159...`	Pi.
`EUL`	`2.71828...`	Euler's constant, also referred to as 'e.'

7.4 Printing

Any type in Odds can be printed. Print, in addition to printing, returns the stringified version of the variable passed to it.

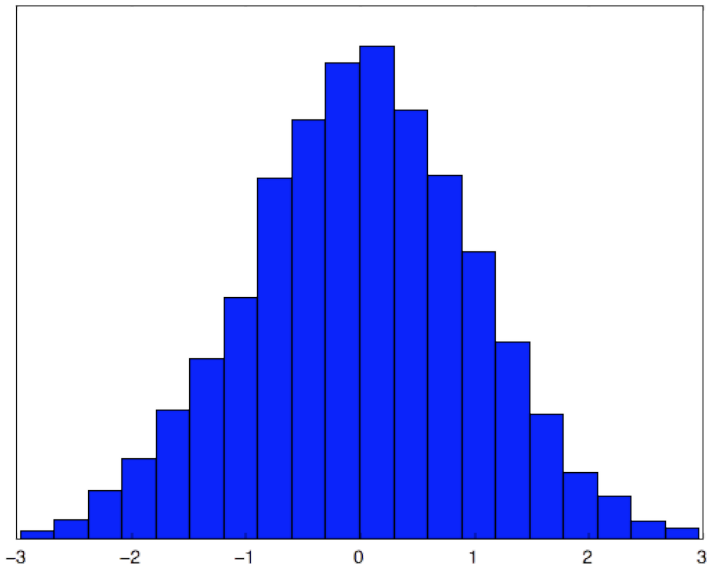
Example:

```
do str_1 = print(1) /* str_1 = "1" */
```

If a distribution is fed to print, it will print a histogram.

```
do norm = <-3, 3> | normal |
do print(norm)
```

Outcome:



8. Project Plan

8.1 Process

8.1.1 Planning

Odds was developed iteratively from start to end. We broke the large project down into several smaller sections - language specifications, scanner, parser, analyzer, generator - and cycled through designing, programming, and testing. Everyone was very involved in this project, and took responsibility for different parts in the compiler. Whenever we started working on a new milestone, we worked independently before we met to combine our individual parts together. We then collaborated together (as a whole group or subgroups) to make sure everyone was on the same page. Before we parted ways, our team divided up future goals to be completed by the next meeting.

8.1.2 Specification

We chose to make *Odds* a functional language for a number of reasons. The core datatype of our language is the distribution type, which is a range of numbers weighted by a function. As a consequence, our users require the ability to pass functions with ease. A functional language seemed most appropriate for this use case. Additionally, because *Odds* centers around numerous mathematical calculations, treating everything as an expression was the most efficient way for our users to make these calculations. Thus, operations which are not traditionally expressions in imperative languages (such as conditionals) can be used seamlessly within a mathematical calculation, improving the flow of code creation.

We also liked the way OCaml code looks, and how short and readable it can be without the type annotations that other languages such as Java and C require. We decided that we wanted our language to be type annotation-less as well. Little did we realize how much of a challenge this seemingly minor aesthetic choice would present. We originally intended for our language to be ‘type ignorant’ like Python (quoth Professor Edwards), but realized halfway through that we ought to add type inference and checking. We hope that this decision and the hours of toil it entailed make *Odds* code not only highly functional, but also succinct and pretty. Finally, we decided to make *Odds* statically scoped (as in OCaml) rather than dynamically scoped (as in Python).

8.1.3 Development

The roadmap for our development was quite straightforward. We decided it was important to finish the whole pipeline first with very simple types - numerals, booleans, strings. Therefore, our first milestone was to make a very basic scanner, parser, and generator. We then added the analyzer for type inferences and semantic analysis.

Finally, we built up the basic pipeline by building in lists and distributions. Whenever we worked on a new feature, we always created a new branch and never pushed straight to master. No pull request was accepted unless it passed all of the tests and if it didn't break the master branch after merging.

8.1.3 Testing

While developing the code, we concurrently tested what we wrote. This meant that after writing our scanner, we set up automated testing to test if our scanner was tokenizing inputs correctly. As we added and removed tokens, we added tests to reflect our updates. This process was repeated for the parser, code generation, and standard library functions. All the testing suite and processes will be discussed in detail later in the testing section of this report.

8.2 Style Guide

While programming, all group members followed these following style guidelines to ensure our project stayed consistent:

- Lines of code should not be more than 80 characters
- No tabs for indentation
- Indentation is always 2 spaces
- Naming consistency between the different program files
- Newline at the end of each file
- One line between each function
- White space for readability

8.3 Timeline

September 23	First Commit
September 30	Submitted Project Proposal
October 14	Finished Initial Scanner
October 18	Automated Testing on Travis CI
October 26	Submitted Language Reference Manual
November 16	Successfully Generated Code ("Hello World")
November 17	Overhauled and Restructured Language

December 2	Successfully Generated Code (with Type Inference and IR)
December 21	Project Presentation and Submission

8.4 Roles and Responsibilities

Initially, we assigned one of the four main roles - *Manager, Language Guru, System Architect, Tester* - to each member on the team. Although each person eventually grew into their own role, the roles started to bleed together as we developed and planned the language. The team frequently coded together when we met, and helped each other when we ran into a particularly difficult problem. The table below illustrates the main roles we split into, and one example of a part we contributed heavily in and/or spearheaded:

Team Member	**Responsibilities**
Alex Kalicki	*System Architect*, Git Guru, Travis' Best Friend
Alexandra Medway	*Manager*, Code Generation
Danny Echikson	*Language Guru*, Semantic Checking and Type Inference
Lilly Wang	*Tester*, Intermediate Code Generation

8.5 Software Development Environment

Programming Language Stack

- **Git Repository (Hosted on Github)** - for version control system that contains the compiler code and test suite
- **Ocaml 4.02.3** - for scanning, parsing, and semantically checking the odds source code and generation of Python target code
- **Python 2.7** - for writing some of the core, built-in functions of our language
- **Bash Shell Scripts** - for running our program given an input odds file (.ods) and an output python file (.py) file, as well as automating testing
- **Makefile** - for all things compiling, linking, and test related

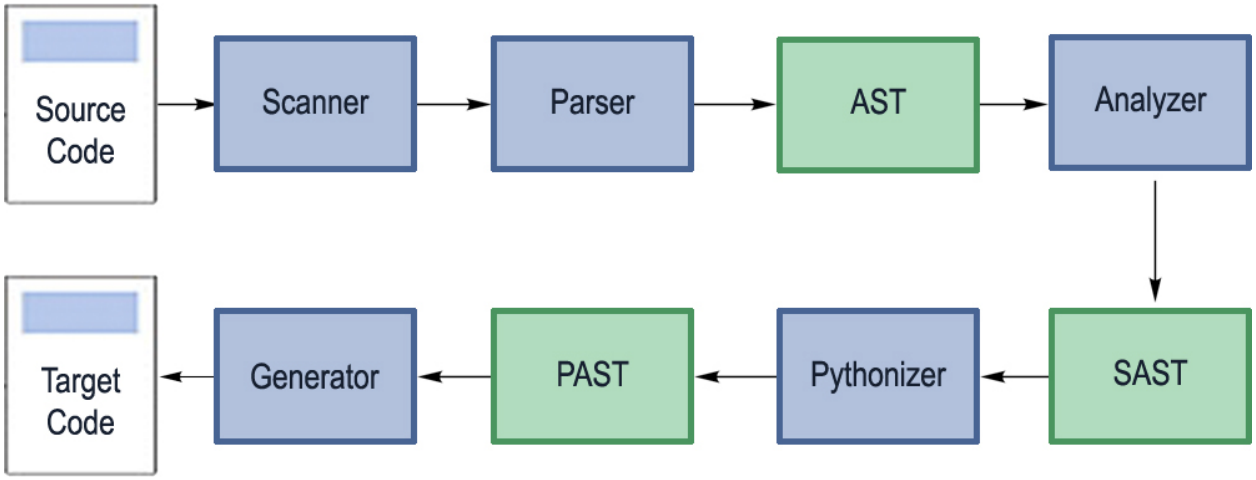
Tools

- **Travis CI** - for automated continuous integration testing through Github to make sure no new code modifies the correct functionality of the language
- **Sublime Text, vim** - for text editing, depending on each team member's preference

9. Architectural Design

9.1 Compiler Overview

The architecture of the *Odds* translator consists of the following major components: scanner, parser, analyzer (type inference and semantic checking), pythonizer (an intermediate representation), and generator (code generation). All of the major components were implemented in OCaml, and the flow of the architecture is shown in the diagram below:



The entry point to our language is through *odds.sh*. This script takes the standard library functions (list.ods and dist.ods) and adds them to the beginning of the source code that needs to be compiled. It then sends the file into *odds.ml*, which sequentially calls the relevant parts of the compiler to generate the target python code. Before it writes the target code to an output file, it adds the built-in functions (core.py) to the translated *Odds* file.

9.2 Scanner

Relevant files: *scanner.mll*

The scanner is written in *OCamlLex* and takes an *Odds* file (.ods). It throws away all whitespace and comments, and it tokenizes the input file into keywords, literals, operators, and identifiers. If a token is not recognized, it throws an error and the *Odds* program is not translated.

The tokens from the scanner are then passed to the parser.

9.3 Parser and AST

Relevant files: *parser.mly, ast.mli*

The tokens from the scanner are taken into the parser and put together to form an abstract syntax tree. The parser is written in *OCamlYacc* and defines the *Odds* grammar by generating a tree using data types described in the abstract syntax tree and precedence rules established in the parser. If there is no rule on how to parse a sequence of tokens, the parser will throw an error and the *Odds* program is not translated.

The AST from the parser is sent to the analyzer for semantic analysis.

9.4 Analyzer and SAST

Relevant files: *analyzer.ml, sast.mli*

The analyzer takes the abstract syntax tree (AST) generated by the parser and checks it for semantic correctness. It recursively traverses the original tree, and at each node builds a new “semantically-checked” node that contains the node’s type information. For example, an AST binary operator “sum” node will be translated to a semantically-checked “sum” node with type Num in the analyzer, and both subtrees of this node will also be constrained to type Num. If the Analyzer attempts to constrain subtrees with invalid types, it throws a Semantic Error.

In addition to performing type inference on *Odds* code and catching type compatibility errors, the analyzer handles the language’s static scoping. At each node in the traversed AST, the analyzer passes an environment variable with information about the information available to the code at that point in time. This allows the analyzer to catch errors such as “variable not found in scope” but also handles code that defines local variables and uses identifiers from a parent scope. When an identifier node is found by the analyzer traversing the AST, the program replaces the identifier with its corresponding statically scoped string so that the python code eventually generated will refer to the correct variables at each point in time.

The analyzer builds a semantically-checked abstract syntax tree (SAST) from the original AST the parser generates and passes the tree to the pythonizer to create an intermediate representation capable of being generated.

9.5 Pythonizer and PAST

Relevant files: *pythonizer.ml, past.mli*

The pythonizer takes in the SAST and is responsible for generating an intermediate representation of the *Odds* translated code. The main functionalities of the pythonizer are to translate the distributions into lists (which is how we are treating distributions under the hood), change some things from expressions to statements, and “pull up” functions and assignments.

In our SAST, the only type of statement is Do(expr). However, Python’s expression tree is a lot different than our language, so we decided to create a PAST (python abstract syntax tree) that is more reflective of our target language’s abstract syntax tree. We had to change the returns, function definitions, conditionals, and assignments from expressions into statements in the pythonizer.

To permit chaining of assignments (e.g. `a = b = 42`), we “pulled up” assignments. We also pulled up functions to take care of the anonymous functions that are allowed in our language (without using Python’s lambda functions). What we mean by “pulling up” is that we added assignments and functions into the statement list before they were originally created and replacing those values later on with the newly created identifiers. This behavior allows for such complicated operations as assignment chaining, anonymous functions, if statements, function declaration equality checks, and many other operations that would not be possible with a direct translation of the original *Odds* code.

The PAST is then sent to the generator, where it is be turned into the translated target code.

9.6 Generator

Relevant files: *generator.ml*

Since a lot of work was taken care of in the PAST, generating Python code from the PAST is incredibly straight-forward and quick. It takes care of the indentation by traversing the tree and keeping track of what level it is on.

The output of this is sent to *odds.ml* which adds the core built-in functions at the top of the translated target code and writes the result in a python file.

10. Test Plan

10.1 Source to target examples

Note that the standard library and built in functions prepended to the generated python files have been truncated to only include the relevant sections. To see the entire standard library (in *Odds* and in Python), as well as the built in functions, see the appendix.

Example 1: Anonymous Functions and Assignment

```
do four_or_two = (val) ->
    do z = y = if val then (x) -> return x + 4 else (x) -> return x + 2
    return z
```

Example 1: Output

```
def four_or_two_0(val_1):
    def _anon_2(x_3):
        return (x_3 + 4)
    def _anon_4(x_5):
        return (x_5 + 2)
    def _cond_6():
        if val_1:
            return _anon_2
        return _anon_4
    y_7 = _cond_6()
    z_8 = y_7
    z_8
    return z_8
four_or_two_0
```

Example 2: Distributions

```
/* flipping a coin to win a dollar*/
do coin = |<[0,1], [.5, .5]>|
/* profit on a normally distributed lottery */
do lot = <-3,3> | normal |
/* expected combined profit */
do combined = coin <+> lot
do E(combined)
```

Example 2: Output

```
from __future__ import print_function, division
import math
import random
import sys
```

```
# Odds constants
EUL = math.e
PI = math.pi

INDEX_STEP = 1000
DIST_LENGTH = 10000
SAMPLE_STEP = 100

def print(*args, **kwargs):
    """Plot distributions for long lists and call normal print() function,
    but return argument that was passed"""
    if type(args[0]) is list and len(args[0]) >= DIST_LENGTH:
        print_dist(args[0])
        return str(args[0])
    __builtins__.print(*args, **kwargs)
    return str(args[0])

def print_dist(dist):
    """Opens a new figure (window) for each distribution it prints, removes
    the y-axis labels, and does not show them all until the end"""
    import matplotlib.pyplot as plt
    global PLOT
    PLOT = True
    plt.figure()
    plt.hist(dist, bins=20, normed=True)
    ax = plt.gca()
    ax.axes.get_yaxis().set_visible(False)

def make_dist(start, end, f):
    """Return a list generated from dist<min, max> | f"""
    if end <= start:
        exception("dist_make: start cannot be greater than end")
    step = (end - start) * 1.0 / INDEX_STEP
    indices = [ start + step * x for x in range(INDEX_STEP) ]

    cum_sum = 0.0
    cum_weights = []
    for x in indices:
        cum_sum += abs(f(x))
        cum_weights.append(cum_sum)
    rands = sorted([ random.uniform(0, cum_sum) for x in range(DIST_LENGTH) ])

    cum_i = 0
    rand_i = 0
    dist_list = []
    while rand_i < len(rands):
        if rands[rand_i] < cum_weights[cum_i]:
            dist_list.append(indices[cum_i])
            rand_i = rand_i + 1
        else:
            cum_i = cum_i + 1
    return dist_list
```

```
def dist_add(d1, d2):
    """Return the sum of two distributions, adding each combination"""
    s1 = d1[random.randint(0, SAMPLE_STEP - 1)::SAMPLE_STEP]
    s2 = d2[random.randint(0, SAMPLE_STEP - 1)::SAMPLE_STEP]
    return sorted([ x + y for x in s1 for y in s2 ])

def E(d):
    """ Expected value of the distribution """
    return sum(d) * 1.0 / DIST_LENGTH

"""
END ODDS CORE LIBRARY
BEGIN USER CODE
"""

def normal_79(x_80):
    coef_81 = (1 / ((2 * PI) ** 0.5))
    coef_81
    exp_82 = (-((1 * (x_80 ** 2)) / 2))
    exp_82
    return (coef_81 * (EUL ** exp_82))
normal_79
coin_83 = make_discr_dist([0, 1], [0.5, 0.5])
coin_83
lot_84 = make_dist((-3), 3, normal_79)
lot_84
combined_85 = dist_add(coin_83, lot_84)
combined_85
E(combined_85)
```

10.2 Test Suite

As we worked on the different sections of the compiler, we made sure to add tests in parallel with what we developed. Our team found this incredibly useful, especially when we had to rewrite and refactor a large section of our code in the middle of the semester. If a test failed, it was very easy to identify where in the compiler the bug was and which commit introduced the bug. From the names of the test files, it is clear to see what we were testing in the following sections (the files are also in the appendix of this report should anyone be curious about the contents of these tests):

10.2.1 Scanner

All .in files are the input test files that need to be tokenized. We wrote .out files of how the input file should be tokenized for each test, and used those to test the output of the scanner.

Since we did not have a parser when writing the scanner, we wrote tokenize.ml so the scanner tests could run independently of the parser.

```
./scanner_test.sh
Running scanner tests...
  - checking scanner/_arithmetic.in...                          SUCCESS
  - checking scanner/_assignment.in...                          SUCCESS
  - checking scanner/_comment.in...                             SUCCESS
  - checking scanner/_conditional.in...                         SUCCESS
  - checking scanner/_declarative.in...                         SUCCESS
  - checking scanner/_dist.in...                                SUCCESS
  - checking scanner/_evenfunc.in...                            SUCCESS
  - checking scanner/_function.in...                            SUCCESS
  - checking scanner/_identifier.in...                          SUCCESS
  - checking scanner/_literal.in...                             SUCCESS
  - checking scanner/_logical.in...                             SUCCESS
  - checking scanner/_mixed_arithmetic_literal.in...            SUCCESS
  - checking scanner/_punctuation.in...                         SUCCESS
  - checking scanner/_relational.in...                          SUCCESS
```

10.2.2 Parser

All .in files are the input test files that need to be parsed. We wrote .out files of how the input file should be parsed for each test, and used those to test the output of the parser.

In order to test the output of the parser, we needed to be able to print the AST that is generated by the parser. We wrote parserize.ml to achieve this.

```
./parser_test.sh
Running parser tests...
  - checking parser/_arithmetic.in...                           SUCCESS
  - checking parser/_cake.in...                                 SUCCESS
  - checking parser/_conditional.in...                          SUCCESS
  - checking parser/_dist.in...                                 SUCCESS
  - checking parser/_func_decl.in...                            SUCCESS
  - checking parser/_id_call.in...                              SUCCESS
  - checking parser/_list.in...                                 SUCCESS
  - checking parser/_literal.in...                              SUCCESS
  - checking parser/_logical.in...                              SUCCESS
  - checking parser/_relational.in...                           SUCCESS
```

10.2.3 Compiler

All .ods files are the input test files that need to be generated. We wrote .py files of how the input file should be generated for each test, and used those to test the output of the generator.

Also, we wrote .out files if it was applicable (if the program printed anything to stdout). We used these files by running the generated python file and comparing the stdout to the corresponding .out file.

```
./pass_test.sh
Running compiler 'pass' tests...
  - checking compiler/pass/_anon_in_scope.ods...                SUCCESS
  - checking compiler/pass/_arithmetic.ods...                   SUCCESS
  - checking compiler/pass/_assign_equality.ods...              SUCCESS
  - checking compiler/pass/_assignment.ods...                   SUCCESS
  - checking compiler/pass/_cake.ods...                         SUCCESS
  - checking compiler/pass/_fdecl.ods...                        SUCCESS
  - checking compiler/pass/_fdecl_anon.ods...                   SUCCESS
  - checking compiler/pass/_fdecl_equality.ods...               SUCCESS
  - checking compiler/pass/_fdecl_nested.ods...                 SUCCESS
  - checking compiler/pass/_hello_world.ods...                  SUCCESS
  - checking compiler/pass/_id_call.ods...                      SUCCESS
  - checking compiler/pass/_if.ods...                           SUCCESS
  - checking compiler/pass/_list.ods...                         SUCCESS
  - checking compiler/pass/_list_ops.ods...                     SUCCESS
  - checking compiler/pass/_literal.ods...                      SUCCESS
  - checking compiler/pass/_logical.ods...                      SUCCESS
  - checking compiler/pass/_relational.ods...                   SUCCESS
./fail_test.sh
Running compiler 'fail' tests...
  - checking compiler/fail/_binop_logical.ods...                SUCCESS
  - checking compiler/fail/_binop_numeric.ods...                SUCCESS
  - checking compiler/fail/_call_func_param_num.ods...          SUCCESS
  - checking compiler/fail/_call_length.ods...                  SUCCESS
  - checking compiler/fail/_call_nonfunc.ods...                 SUCCESS
  - checking compiler/fail/_call_types.ods...                   SUCCESS
  - checking compiler/fail/_discr_dist.ods...                   SUCCESS
  - checking compiler/fail/_fdecl_anon.ods...                   SUCCESS
  - checking compiler/fail/_fdecl_nested.ods...                 SUCCESS
  - checking compiler/fail/_if_cond.ods...                      SUCCESS
  - checking compiler/fail/_if_mismatch.ods...                  SUCCESS
  - checking compiler/fail/_illegal_char.ods...                 SUCCESS
  - checking compiler/fail/_list_head_types.ods...              SUCCESS
  - checking compiler/fail/_list_heterogeneous.ods...           SUCCESS
  - checking compiler/fail/_list_tail_types.ods...              SUCCESS
  - checking compiler/fail/_print_return_type.ods...            SUCCESS
  - checking compiler/fail/_rec_param_change.ods...             SUCCESS
  - checking compiler/fail/_rec_return_change.ods...            SUCCESS
  - checking compiler/fail/_unop_not.ods...                     SUCCESS
  - checking compiler/fail/_unop_sub.ods...                     SUCCESS
  - checking compiler/fail/_var_scope.ods...                    SUCCESS
```

10.2.4 Standard Library Functions

All .ods files are the input test files that need to be generated. We wrote .out files of expected output on stdout when running the compiled Python files, and used those to test the output of the generator.

These tests make sure the standard library and built in functions work as they should. Since the standard library is always be added to the top of the .py files, we knew that all our previous tests for the scanner, parser, and compiler would fail on direct Python comparison. Therefore, we added a flags to our odds.sh file to let the user indicate if they wanted to compile it raw ("`-r`") without the standard libary or normal ("`-c`") with the standard library. We ran the scanner, parser, and compiler tests with the `-r` flag and the library tests with the `-c` flag.

```
./lib_test.sh
Running compiler 'lib' tests...
  - checking lib/_discr_dist.ods...                                SUCCESS
  - checking lib/_dist.ods...                                      SUCCESS
  - checking lib/_dist_lib.ods...                                  SUCCESS
  - checking lib/_exception.ods...                                 SUCCESS
  - checking lib/_list_concat.ods...                               SUCCESS
  - checking lib/_list_empty.ods...                                SUCCESS
  - checking lib/_list_fold.ods...                                 SUCCESS
  - checking lib/_list_get.ods...                                  SUCCESS
  - checking lib/_list_get_error.ods...                            SUCCESS
  - checking lib/_list_insert.ods...                               SUCCESS
  - checking lib/_list_insert_error.ods...                         SUCCESS
  - checking lib/_list_iter.ods...                                 SUCCESS
  - checking lib/_list_make.ods...                                 SUCCESS
  - checking lib/_list_map.ods...                                  SUCCESS
  - checking lib/_list_remove.ods...                               SUCCESS
  - checking lib/_list_remove_error.ods...                         SUCCESS
  - checking lib/_list_rev.ods...                                  SUCCESS
  - checking lib/_pi_eul.ods...                                    SUCCESS
  - checking lib/_print_return.ods...                              SUCCESS
```

10.3 Test Automation

To test our code, we had several makefiles that would run the scanner, parser, compiler, and library tests after typing in `make test`. It calls the different shell scripts that tests the 100+ test files we created over the duration of this project.

In addition to testing it on our own computers and then pushing our code to Github, we used Travis CI to test our code on Github. Travis became a beloved fifth member of our group, and he prevented us from ever making a bad push onto master. He would automatically invoke `make test` and create builds for every branch commit as well as every created pull request. We set the permissions on Github so that we could not push directly to master and so that any pull request can only be merged once it passed all

tests (before and after merging). As a result of this workflow, we never had to worry about anything breaking the master branch.

11. Lessons Learned

11.1 Alex Kalicki

This project taught me in a very big way that finding a process and a team that works well is half the battle. I credit a huge part of our group's success and final product to our excellent communication, clear workflow and testing, and willingness to take responsibility for distinct portions of the project and see them through to fruition. I learned that a good design makes good code almost write itself, and a bad one can leave you puzzling over a small problem for days. I was happy to be part of a group where each member had strong opinions and would defend them passionately at every turn. I learned that a strong management style (go Alexandra!) and regular meetings are incredibly important for a project of this scale. I also found it very helpful to maintain a strong working knowledge of all parts of the project, and looked forward to our "merge meetings" where we'd all discuss what we had written in the last week.

11.2 Alexandra Medway

I learned that working with a group can be a very rewarding experience. Collaboration and communication are the keys to creating a truly extraordinary product. Source control is the best way to optimize a group project, while dividing and conquering is the best way to optimize the group's time. Spend time on design, and you'll save time on implementation and expansion in later steps. I also learned how important it is to document code; when working in a group setting, everyone's code should be extremely transparent. Otherwise, thought processes appear cryptic, and you'll spend more time trying to understand a section of code than you will trying to add in your feature, or fix a bug.

11.3 Danny Echikson

I learned that the best way to solve a bug in your compiler is to give your group members quesadillas. Kidding? This is the biggest programming project I have ever worked on, and I learned quite a bit in addition to the quesadilla debugging technique. Dealing with people is dissimilar to dealing with computers (surprise!), but dealing with people is far more rewarding than dealing with computers. Though often I do not look forward to slogging through mounds of homework, it certainly becomes more bearable with others. I was also largely responsible for the type inference/type checking portion of our compiler. It was tough. For every two steps forward - figuring out the conditions

under which I might constrain a type, fixing an odd corner case only some evil programmer could have dreamt of, etc. - it was one and a half steps backwards - that solution caused a cascade of other problems that need to be resolved. Programming can be about smarts and patient planning, but it can also be about tenacity.

11.4 Lilly Wang

I learned that your teammates can really make or break your experience in a group project. Having good team members catapulted this class into the top slot of “best class in college” in my mind (which was previously held by Biology… which is pretty blasphemous as a CS major). When you’re stuck on something or think that there is no solution to a problem, having more than one brain thinking about it makes a massive difference. I also learned how to use Github in a really fun, collaborative matter (read: how to insert a GIF in markdown). The notion of “too many GIFs” does not exist in the world of Github.

12. Appendix

12.1 Compiler Code

odds.sh

```
#!/bin/bash

MYDIR="$(dirname "$(which "$0")")"
ODDS_FILE="$MYDIR/compiler/odds"
LIST_LIB="$MYDIR/compiler/lib/list.ods"
DIST_LIB="$MYDIR/compiler/lib/dist.ods"
PY_LIB="$MYDIR/compiler/lib/core.py"

if [ ! -f $ODDS_FILE ]; then
    printf "ERROR: not yet compiled, run 'make' first.\n" 1>&2
    exit 1
fi

# odds.sh (-c | -r) <odds_file> <output_file>
if [ "$#" -eq 3 ]; then
    if [ "$1" == "-c" ]; then
        cat $LIST_LIB $DIST_LIB $2 | $ODDS_FILE $1 $3 $PY_LIB
        exit 0
    elif [ "$1" == "-r" ]; then
        $ODDS_FILE $1 $3 < $2
        exit 0
    else
        printf "ERROR: invalid arguments supplied for command $0 $1\n" 1>&2
        exit 1
    fi
fi

# odds.sh -s <odds_file>
if [ "$#" -eq 2 ] && [ "$1" == "-s" ]; then
    $ODDS_FILE $1 < $2
    exit 0
fi

# odds.sh -h
if [ "$#" -eq 1 ] && [ "$1" == "-h" ]; then
    $ODDS_FILE -h
    exit 0
fi

printf "ERROR: invalid arguments. Run $0 -h for usage instructions\n" 1>&2
exit 1
```

odds.ml

```
(*
 * COMS4115: Odds abstract syntax tree
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

open Printf

type action = Compile | Help | Raw | Sast

let get_help =
  "Odds Usage: odds.sh <flag> [input_file] [output_file]\n" ^
  "  -c\tCompile odds input_file to python code in output_file with stdlib\n" ^
  "  -h\tDisplay this list of options\n" ^
  "  -r\tCompile odds input_file into raw python output_file\n" ^
  "  -s\tPrint odds input_file as semantically checked ast"

(* Error reporting helper function *)
let get_pos_and_tok lexbuf =
  let cur = lexbuf.Lexing.lex_curr_p in
  let line_num = cur.Lexing.pos_lnum and
    column_num = cur.Lexing.pos_cnum - cur.Lexing.pos_bol and
    token = Lexing.lexeme lexbuf in
  line_num, column_num, token

let _ =
  let action = List.assoc Sys.argv.(1)
    [("-c", Compile) ; ("-h", Help) ; ("-r", Raw); ("-s", Sast)] in
  if action = Help then print_endline get_help else
  let lexbuf = Lexing.from_channel stdin in
  try
    let ast = Parser.program Scanner.token lexbuf in
    let sast = Analyzer.check_ast ast in
    let past = Pythonizer.generate_past sast in
    let prog = Generator.gen_program past in
    match action with
      | Compile ->
          let output_file = Sys.argv.(2) and stdlib_file = Sys.argv.(3) in
          let stdlib = Utils.str_of_file stdlib_file in
          let file = open_out output_file
          in fprintf file "%s\n\n%s\n%s\n"
            stdlib prog (Utils.conclude_program ()); close_out file
      | Raw ->
          let output_file = Sys.argv.(2) in
          let file = open_out output_file
          in fprintf file "%s\n" prog; close_out file
      | Sast -> Printer.print_sast sast
```

```
      | Help -> print_endline get_help
  with
    | Scanner.Illegal_Character(m) ->
        let line_num, column_num, _ = get_pos_and_tok lexbuf in
        eprintf
          "\x1b[31mSyntax error\x1b[0m, line %d at column %d: %s\n"
          line_num column_num m
    | Analyzer.Semantic_Error(m) ->
        eprintf "\x1b[31mSemantic error\x1b[0m:\n  %s\n" m
    | Parsing.Parse_error ->
        let line_num, column_num, token = get_pos_and_tok lexbuf in
        eprintf
        "\x1b[31mSyntax error\x1b[0m, line %d at column %d: '%s'\n"
        line_num column_num token
```

scanner.mll

```
(*
 * COMS4115: Odds scanner
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

{
  open Parser
  exception Illegal_Character of string
}

let numeric = ['0'-'9']
let whitespace = [' ' '\n' '\r' '\t']
let newline = '\n' | "\r\n"

rule token = parse

(* Newline - for line number on error report to user *)
| newline   { Lexing.new_line lexbuf; token lexbuf }

(* Whitespace *)
| whitespace    { token lexbuf }

(* Comments *)
| "/*"    { comment lexbuf }

(* Function Symbols & Keywords *)
| ')' whitespace* "->"   { FDELIM }  | "return"   { RETURN }
| ')' whitespace* '('    { CAKE }

(* Punctuation *)
```

```
| '('   { LPAREN }  | ')'   { RPAREN }
| '>' whitespace* '|'       { DDELIM }
| '|' whitespace* '<'       { DISC }
| '<'   { LCAR }    | '>'   { RCAR } (* Also relational operators *)
| '['   { LBRACE }  | ']'   { RBRACE }
| ','   { COMMA }   | '|'   { VBAR }

(* Dist Operators *)
| "<+>" { DPLUS }   | "<*>" { DTIMES }
| "|**" { DPOWER }  | "|*"  { DSTRETCH }
| "|+"  { DSHIFT }

(* List Operator *)
| "::"    { CONS }

(* Arithmetic Operators *)
| '+'   { PLUS }    | '-'   { MINUS }
| '*'   { TIMES }   | '/'   { DIVIDE }
| '%'   { MOD }     | "**"  { POWER }

(* Relational Operators *)
| "=="    { EQ }    | "!="    { NEQ }
| "<="    { LEQ }   | ">="    { GEQ }

(* Logical Operators & Keywords*)
| "&&"    { AND }   | "||"    { OR }
| "!"     { NOT }

(* Assignment Operator *)
| '='   { ASN }

(* Conditional Operators *)
| "if"    { IF }    | "then"  { THEN }
| "else"  { ELSE }

(* Declarative Keywords *)
| "do"    { DO }

(* Literals *)
| numeric+ as intlit { NUM_LITERAL(Ast.Num_int(int_of_string intlit)) }
| numeric* '.' numeric+ as floatlit
    { NUM_LITERAL(Ast.Num_float(float_of_string floatlit)) }
| '"' (([^ '"'] | "\\\"")* as strlit) '"' { STRING_LITERAL(strlit) }
| "true" | "false" as boollit { BOOL_LITERAL(bool_of_string boollit)}
| "void" { VOID_LITERAL }

(* Identifiers *)
| ['a'-'z' 'A'-'Z'] (['a'-'z' 'A'-'Z' '_' ] | numeric)* as lxm { ID(lxm) }

(* End-of-File *)
| eof { EOF }

(* Invalid Token *)
```

```
| _ as char {
    let message = "illegal character '" ^ Char.escaped char ^ "'" in
    raise (Illegal_Character message)
  }

and comment = parse
| "*/"    { token lexbuf }
| newline { Lexing.new_line lexbuf; comment lexbuf }
| _       { comment lexbuf }
```

parser.mly

```
/*
 * COMS4115: Odds parser
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 */

%{ open Ast %}

/* Punctuation */
%token LPAREN RPAREN LCAR RCAR LBRACE RBRACE COMMA VBAR DDELIM DISC

/* Arithmetic Operators */
%token PLUS MINUS TIMES DIVIDE MOD POWER DPLUS DTIMES DPOWER DSHIFT DSTRETCH

/* List Operators */
%token CONS

/* Relational Operators */
%token EQ NEQ LEQ GEQ

/* Logical Operators & Keywords*/
%token AND OR NOT

/* Assignment Operator */
%token ASN

/* Conditional Operators */
%token IF THEN ELSE

/* Declarative Keywords */
%token DO

/* Function Symbols & Keywords */
%token FDELIM RETURN CAKE

/* End-of-File */
```

```
%token EOF

/* Identifiers */
%token <string> ID

/* Literals */
%token <Ast.num> NUM_LITERAL
%token <string> STRING_LITERAL
%token <bool> BOOL_LITERAL
%token VOID_LITERAL

/* Precedence and associativity of each operator */
%nonassoc RETURN
%right ASN
%right ELSE
%left CONS
%left OR
%left AND
%right NOT
%left LCAR LEQ RCAR GEQ EQ NEQ
%left DSHIFT DSTRETCH
%left PLUS MINUS DPLUS
%left TIMES DIVIDE MOD DTIMES
%left DPOWER
%left POWER

%start program                /* Start symbol */
%type <Ast.program> program   /* Type returned by a program */

%%

/* Program flow */
program:
  | stmt_list EOF                         { List.rev $1 }

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

stmt:
  | DO expr                               { Do($2) }

/* Expressions */
expr:
  | literal                               { $1 }
  | arith_ops                             { $1 }
  | bool_ops                              { $1 }
  | list_ops                              { $1 }
  | dist_ops                              { $1 }
  | dist                                  { Dist($1) }
  | discr_dist                            { Discr_dist($1) }
  | ID                                    { Id($1) }
  | ID ASN expr                           { Assign($1, $3) }
```

```
  | ID LPAREN list_opt RPAREN             { Call(Id($1), $3) }
  | LBRACE list_opt RBRACE                { LDecl($2) }
  | LPAREN expr RPAREN                    { $2 }
  | fdecl                                 { Fdecl($1) }
  | LPAREN fdecl CAKE list_opt RPAREN     { Cake(Fdecl($2), $4) }
  | IF expr THEN expr ELSE expr           { If($2, $4, $6) }

/* Function declaration */
fdecl:
  | LPAREN fparam_opt FDELIM stmt_list RETURN expr
    { {
      params = $2;
      body = List.rev $4;
      return = $6;
    } }

fparam_opt:
  | /* nothing */                         { [] }
  | fparam_list                           { List.rev $1 }

fparam_list:
  | ID                                    { [$1] }
  | fparam_list COMMA ID                  { $3 :: $1 }

/* Lists and function calling */
list_opt:
  | /* nothing */                         { [] }
  | list                                  { List.rev $1 }

list:
  | expr                                  { [$1] }
  | list COMMA expr                       { $3 :: $1 }

/* Distributions */
dist:
  | LCAR expr COMMA expr DDELIM expr VBAR
    { {
      min = $2;
      max = $4;
      dist_func = $6;
    } }

discr_dist:
  | DISC expr COMMA expr DDELIM
    { {
      vals = $2;
      weights = $4;
    } }

/* Binary operators */
arith_ops:
  | MINUS expr                            { Unop(Sub, $2) }
  | expr PLUS expr                        { Binop($1, Add, $3) }
```

```
  | expr MINUS expr                       { Binop($1, Sub, $3) }
  | expr TIMES expr                       { Binop($1, Mult, $3) }
  | expr DIVIDE expr                      { Binop($1, Div, $3) }
  | expr MOD expr                         { Binop($1, Mod, $3) }
  | expr POWER expr                       { Binop($1, Pow, $3) }

bool_ops:
  | NOT expr                              { Unop(Not, $2) }
  | expr OR expr                          { Binop($1, Or, $3) }
  | expr AND expr                         { Binop($1, And, $3) }
  | expr EQ expr                          { Binop($1, Eq, $3) }
  | expr NEQ expr                         { Binop($1, Neq, $3) }
  | expr LCAR expr                        { Binop($1, Less, $3) }
  | expr LEQ expr                         { Binop($1, Leq, $3) }
  | expr RCAR expr                        { Binop($1, Greater, $3) }
  | expr GEQ expr                         { Binop($1, Geq, $3) }

dist_ops:
  | expr DPLUS expr                       { Binop($1, D_Plus, $3) }
  | expr DTIMES expr                      { Binop($1, D_Times, $3) }
  | expr DPOWER expr                      { Binop($1, D_Power, $3) }
  | expr DSHIFT expr                      { Binop($1, D_Shift, $3) }
  | expr DSTRETCH expr                    { Binop($1, D_Stretch, $3) }
  | expr LCAR RCAR expr                   { Binop($1, D_Sample, $4)}

list_ops:
  | expr CONS expr                        { Binop($1, Cons, $3) }

/* Literals */
literal:
  | NUM_LITERAL                           { Num_lit($1) }
  | STRING_LITERAL                        { String_lit($1) }
  | BOOL_LITERAL                          { Bool_lit($1) }
  | VOID_LITERAL                          { Void_lit }
```

ast.mli

```
(*
 * COMS4115: Odds abstract syntax tree
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

(* Unary operators *)
type unop =
  | Sub      (* - *)
  | Not      (* ! *)
```

```
(* Binary operators *)
type binop =
  (* Dist *)
  | D_Plus     (* <+> *)
  | D_Times    (* <*> *)
  | D_Power    (* |** *)
  | D_Shift    (* |+ *)
  | D_Stretch  (* |* *)
  | D_Sample   (* <> *)
  (* Arithmetic *)
  | Add      (* + *)
  | Sub      (* - *)
  | Mult     (* * *)
  | Div      (* / *)
  | Mod      (* % *)
  | Pow      (* ** *)
  (* Boolean *)
  | Or       (* || *)
  | And      (* && *)
  | Eq       (* == *)
  | Neq      (* != *)
  | Less     (* < *)
  | Leq      (* <= *)
  | Greater  (* > *)
  | Geq      (* >= *)
  (* List *)
  | Cons     (* :: *)

(* Expressions *)
type num =
  | Num_int of int      (* 42 *)
  | Num_float of float  (* 42.0 *)

type expr =
  | Num_lit of num                (* 42 *)
  | String_lit of string          (* "Hello, world" *)
  | Bool_lit of bool              (* true *)
  | Void_lit                      (* void *)
  | Unop of unop * expr           (* -5 *)
  | Binop of expr * binop * expr  (* a + b *)
  | Id of string                  (* x *)
  | Assign of string * expr       (* x = 4 *)
  | Call of expr * expr list      (* add(1, 2) *)
  | LDecl of expr list            (* [1, 2, 3] *)
  | Dist of dist                  (* < 1, 2> | normal *)
  | Discr_dist of discr_dist      (* |< l1, l2 >| *)
  | Fdecl of fdecl                (* (x) -> ... return x *)
  | Cake of expr * expr list      (* (() -> return 42)() *)
  | If of expr * expr * expr      (* if true then 42 else 43 *)

(* Distribution Declarations *)
and dist = {
  min: expr;        (* Distribution Minimum *)
```

```
  max: expr;        (* Distribution Maximum *)
  dist_func: expr;  (* Distribution Function *)
}

and discr_dist = {
  vals: expr;
  weights: expr;
}

(* Function Declarations *)
and fdecl = {
  params: string list;  (* Parameters *)
  body: stmt list;      (* Function Body *)
  return: expr;         (* Return *)
}

(* Statements *)
and stmt =
  | Do of expr     (* set foo = bar + 3 *)

(* Program entry point *)
type program = stmt list
```

analyzer.ml

```
(*
 * COMS4115: Semantic Analyzer
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

open Ast
open Sast
open Printf

(********************
 * Environment
 ********************)

module VarMap = Map.Make(String)

type environment = {
  params: Sast.var VarMap.t;
  scope: Sast.var VarMap.t;
}

(* Builtin variables and functions *)
let builtins = VarMap.empty
```

```
let builtins = VarMap.add "EUL" { name = "EUL"; s_type = Num; builtin = true; }
builtins
let builtins = VarMap.add "PI" { name = "PI"; s_type = Num; builtin = true; }
builtins

(* Core functions *)
let builtins = VarMap.add "exception" {
  name = "exception";
  s_type = Func({ param_types=[String]; return_type = Void; });
  builtin = true;
} builtins

let builtins = VarMap.add "print" {
  name = "print";
  s_type = Func({ param_types = [Any]; return_type = String; });
  builtin = true;
} builtins

(* Dist builtins *)
let builtins = VarMap.add ".dist_add" {
  name = ".dist_add";
  s_type = Func({ param_types = [Dist_t; Dist_t]; return_type = Dist_t; });
  builtin = true;
} builtins

let builtins = VarMap.add ".dist_mult" {
  name = ".dist_mult";
  s_type = Func({ param_types = [Dist_t; Dist_t]; return_type = Dist_t; });
  builtin = true;
} builtins

let builtins = VarMap.add ".dist_shift" {
  name = ".dist_shift";
  s_type = Func({ param_types = [Num; Dist_t]; return_type = Dist_t; });
  builtin = true;
} builtins

let builtins = VarMap.add ".dist_stretch" {
  name = ".dist_stretch";
  s_type = Func({ param_types = [Num; Dist_t]; return_type = Dist_t; });
  builtin = true;
} builtins

let builtins = VarMap.add ".dist_exp" {
  name = ".dist_exp";
  s_type = Func({ param_types = [Num; Dist_t]; return_type = Dist_t; });
  builtin = true;
} builtins

let builtins = VarMap.add ".dist_sample" {
  name = ".dist_sample";
  s_type = Func({ param_types = [Num; Dist_t]; return_type = List(Num); });
  builtin = true;
```

```
} builtins

let builtins = VarMap.add "P" {
  name = "P";
  s_type = Func({ param_types = [Num; Dist_t]; return_type = Num; });
  builtin = true;
} builtins

let builtins = VarMap.add "E" {
  name = "E";
  s_type = Func({ param_types = [Dist_t]; return_type = Num; });
  builtin = true;
} builtins

(* List builtins *)
let builtins = VarMap.add "head" {
  name = "head";
  s_type = Func({ param_types = [List(Any)]; return_type = Any; });
  builtin = true;
} builtins

let builtins = VarMap.add "tail" {
  name = "tail";
  s_type = Func({ param_types = [List(Any)]; return_type = List(Any); });
  builtin = true;
} builtins

let builtins = VarMap.add ".cons" {
  name = ".cons";
  s_type = Func({ param_types = [Any ; List(Any)]; return_type = List(Any); });
  builtin = true;
} builtins

let builtins = VarMap.add "len" {
  name = "len";
  s_type = Func({ param_types = [List(Any)]; return_type = Num; });
  builtin = true;
} builtins

(* Program entry environment *)
let root_env = {
  params = VarMap.empty;
  scope = builtins;
}

(********************
 * Utilities
 ********************)

(* Given an ssid my_var_#, return the original key ID my_var *)
let id_of_ssid ssid =
  let id_len =
    try String.rindex ssid '_' with Not_found -> String.length ssid in
```

```
  String.sub ssid 0 id_len

let rec str_of_type = function
  | Num -> "Num"
  | String -> "String"
  | Bool -> "Bool"
  | Void -> "Void"
  | List(l) -> sprintf "List[%s]" (str_of_type l)
  | Func(f) -> str_of_func f
  | Dist_t -> "Dist"
  | Any -> "Any"
  | Unconst -> "Unconst"

and str_of_func f =
  let param_types = List.map str_of_type f.param_types and
    return_type = str_of_type f.return_type in
  sprintf "Func(%s => %s)" (String.concat ", " param_types) return_type

let str_of_unop = function
  | Not -> "!"      | Sub -> "-"

let str_of_binop = function
  (* Dist *)
  | D_Plus -> "<+>" | D_Times -> "<*>"
  | D_Shift -> "|+" | D_Stretch -> "|*"
  | D_Sample -> "<>"| D_Power -> "|**"
  (* Arithmetic *)
  | Add -> "+"      | Sub -> "-"
  | Mult -> "*"     | Div -> "/"
  | Mod -> "%"      | Pow -> "**"
  (* Boolean *)
  | Or -> "||"      | And -> "&&"
  | Eq -> "=="      | Neq -> "!="
  | Less -> "<"     | Leq -> "<="
  | Greater -> ">"  | Geq -> ">="
  (* List *)
  | Cons -> "::"

let is_sugar = function
  | Cons | D_Plus | D_Times | D_Shift | D_Stretch | D_Power | D_Sample -> true
  | _ -> false

(* for debugging *)
let print_env env =
  let print_var id var =
    let line = sprintf "\t%s --> { name: %s; s_type: %s; }"
    id var.name (str_of_type var.s_type) in
    print_endline line in
  let str_of_varmap name vm =
    let header = sprintf "%s:" name in
    print_endline header; VarMap.iter print_var vm in
  print_endline "";
  str_of_varmap "env params" env.params;
```

```
  str_of_varmap "env scope" env.scope

(********************
 * Exceptions
 ********************)

exception Semantic_Error of string
exception Collect_Constraints_Error

let var_error id =
  let message = sprintf "Variable '%s' is undefined in current scope" id
  in raise (Semantic_Error message)

let unop_error op t =
  let message = sprintf "Invalid use of unary operator '%s' with type %s"
    (str_of_unop op) (str_of_type t) in
  raise (Semantic_Error message)

let typ_mismatch t1 t2 =
  let message = sprintf "Expected type %s but got type %s instead"
    (str_of_type t1) (str_of_type t2) in
  raise (Semantic_Error message)

let binop_error t1 op t2 =
  let message =
    sprintf "Invalid use of binary operator '%s' with types %s and %s"
    (str_of_binop op) (str_of_type t1) (str_of_type t2) in
  raise (Semantic_Error message)

let fcall_nonid_error () =
  let message = "Sast.Call provided non-ID as first argument" in
  raise (Semantic_Error message)

let fcall_nonfunc_error id typ =
  let id = match id with
    | Sast.Id(ssid) -> id_of_ssid ssid
    | _ -> fcall_nonid_error () in
  let message = sprintf "Attempting to call %s type '%s' as a function"
    (str_of_type typ) id in
  raise (Semantic_Error message)

let fcall_length_error id num_params num_args =
  let name = match id with
    | Sast.Id(name) -> id_of_ssid name
    | _ -> fcall_nonid_error () in
  let message = sprintf
    "Function '%s' expects %d argument(s) but was called with %d instead"
    name num_params num_args in
  raise (Semantic_Error message)

let fcall_argtype_error id typ const =
  let name = match id with
    | Sast.Id(name) -> id_of_ssid name
```

```
    | _ -> fcall_nonid_error () in
  let message = sprintf
    "Function '%s' expected argument of type %s but was passed %s instead"
    name (str_of_type const) (str_of_type typ) in
  raise (Semantic_Error message)

let list_error list_type elem_type =
  let message = sprintf "Invalid element of type %s in list of type %s"
    (str_of_type elem_type) (str_of_type list_type) in
  raise (Semantic_Error message)

let type_mismatch_error id typ const =
  let message = sprintf
    "Invalid usage of id '%s' with type %s when type %s was expected"
    id (str_of_type typ) (str_of_type const) in
  raise (Semantic_Error message)

let return_type_mismatch_error func_id typ1 typ2 =
  let message = sprintf "Invalid return type in function '%s':
    type '%s' expected to be returned, but type '%s' returned instead."
    func_id (str_of_type typ1) (str_of_type typ2) in
  raise (Semantic_Error message)

let fdecl_unconst_error id =
  let message = sprintf
    "Invalid declaration of function '%s' with unconstrained return value" id in
  raise (Semantic_Error message)

let fdecl_reassign_error id typ =
  let message = sprintf
    "Invalid attempt to reassign function identifier '%s' to type %s"
    id (str_of_type typ) in
  raise (Semantic_Error message)

let list_cons_mismatch_error typ const =
  let message = sprintf
    "Invalid attempt to prepend a value of type %s to list of type %s"
   (str_of_type typ) (str_of_type const) in
  raise (Semantic_Error message)

let constrain_error old_type const =
  let message = sprintf "Invalid attempt to change unconstrained type %s to %s"
    (str_of_type old_type) (str_of_type const) in
  raise (Semantic_Error message)

let if_mismatch_error typ1 typ2 =
  let message = sprintf
    "Invalid attempt to use conditional with mismatched types %s and %s"
    (str_of_type typ1) (str_of_type typ2) in
  raise (Semantic_Error message)

let dead_code_path_error ocaml_func_name =
  let message =
```

```
    sprintf "ERROR: DEAD CODE PATH REACHED IN FUNCTION: %s" ocaml_func_name in
  raise (Semantic_Error message)

let invalid_dist_min_max_error typ1 typ2 =
  let message = sprintf "Invalid distribution with min type '%s' and max type '%s'"
    (str_of_type typ1) (str_of_type typ2) in
  raise (Semantic_Error message)

let invalid_discr_dist_error typ1 typ2 =
  let message = sprintf "Invalid distribution with vals type '%s' and weights type
'%s'"
    (str_of_type typ1) (str_of_type typ2) in
  raise (Semantic_Error message)

let invalid_dist_func_type_error invalid_typ f_typ =
  let message = sprintf "Invalid distribution with function '%s'
    (distribution's must have function of type '%s')"
    (str_of_type invalid_typ) (str_of_type f_typ) in
  raise (Semantic_Error message)

let dead_code_path_error ocaml_func_name =
  let message =
    sprintf "ERROR: DEAD CODE PATH REACHED IN FUNCTION: %s" ocaml_func_name in
  raise (Semantic_Error message)

(********************
 * Scoping
 ********************)

(* Variable counter to prevent naming conflicts *)
let ss_counter = ref (-1)

(* Given a string x, get a unique id x_# to use as the next variable *)
let get_ssid name =
  ss_counter := !ss_counter + 1;
  sprintf "%s_%d" name !ss_counter

(* Add 'id' with type 's_type' to the environment scope *)
let add_to_scope env id s_type =
  let ss_id = get_ssid id in
  let var = { name = ss_id; s_type = s_type; builtin = false; } in
  let env' = {
    params = env.params;
    scope = VarMap.add id var env.scope;
  } in
  env', ss_id

(*
 * Add param with 'id' and type Unconst to the environment params, erasing it
 * from the environment scope
 *)
let add_to_params env id =
  let ss_id = get_ssid id in
```

```
  let var = { name = ss_id; s_type = Unconst; builtin = false; } in
  let env' = {
    params = VarMap.add id var env.params;
    scope = VarMap.remove id env.scope;
  } in
  env', ss_id


(**********************************
 * Type inference and constraining
 **********************************)

(* Update the type for given id corresponding to given 'ssid' in env *)
let update_type env ssid typ =
  let id = id_of_ssid ssid in
  if VarMap.mem id env.scope then (VarMap.find id env.scope).s_type <- typ else
  if VarMap.mem id env.params then (VarMap.find id env.params).s_type <- typ
  else var_error id

(*
 * Attempt to constrain an ID in an expression one level down. E.g. !x would
 * constrain x to a boolean and x + y would constrain both x and y to integers,
 * but !(x == y) would not constrain either variable.
 *
 * Takes the current environment, type to constrain, and an expression wrapper
 * in which to search for an ID. Returns the newly constrained environment and
 * expression wrapper on success, or their old values on failure.
 *)
let rec constrain_ew env ew typ =
  let Sast.Expr(e, old_typ) = ew in
  if not (has_unconst old_typ) && old_typ <> typ then constrain_error old_typ typ
else
  match e with
    | Sast.Id(ssid) -> update_type env ssid typ; env, Sast.Expr(e, typ)
    | Sast.Fdecl(f) -> update_type env f.f_name typ; env, Sast.Expr(e, typ)
    | Sast.Call(Sast.Expr(Sast.Id(ssid), Sast.Func(f)), _) ->
        let _, Sast.Expr(_, old_type) = check_id env (id_of_ssid ssid) in
        let old_ret_type = match old_type with
          | Sast.Func(old_f) -> old_f.return_type
          | _ as typ -> fcall_nonfunc_error (Sast.Id(ssid)) typ in
        if not (has_unconst f.return_type) && f.return_type <> old_ret_type then
            constrain_error old_ret_type f.return_type
        else
          if has_any f.return_type then env, Sast.Expr(e, Func(f)) else
            let f' = Func({ f with return_type = typ }) in
          update_type env ssid f'; env, Sast.Expr(e, f')
    | _ -> env, ew

(* This function is the same as constrain_ew, except instead of constraining
 * expression_wrappers, it constrains expressions. This function only modifies
 * the env and does not return an expression wrapper.
 *)
and constrain_e env e typ = match e with
```

```
  | Sast.Id(ssid) -> update_type env ssid typ; env
  | _ -> env

(* This function takes 2 types. It returns 2 types. The first type returned
 * will overwrite 'Any' to another type, including, possibly, 'Unconst.' The
 * second type returned will have 'Any' in it, overwriting any other type
 * when neccessary. TO DO: MAKE PRETTIER & SHORTER.
 *)
and collect_constraints typ1 typ2 =
  (* Helper functions for collect_constraints *)
  let build_func collect_func func1 func2 =
    let params1 = func1.param_types and params2 = func2.param_types and
      ret1 = func1.return_type and ret2 = func2.return_type in
    (* If different number of params, raise error *)
    let params' =
      if List.length params1 <> List.length params2 then
        raise Collect_Constraints_Error
      else List.map2 collect_func params1 params2 and
      ret' = collect_func ret1 ret2 in
    Func({ param_types = params'; return_type = ret'; })
  and build_list collect_func l_typ1 l_typ2 =
    let l_typ' = collect_func l_typ1 l_typ2 in List(l_typ') in

  (* Collects possible constraints and returns type that is as constrained as
   * possible. Any is always converted to Unconst. *)
  let rec overwrite_any typ1 typ2 = match typ1 with
    | Any | Unconst -> any_to_unconst typ2
    | Func(func1) ->
        begin match typ2 with
          | Any | Unconst -> typ1
          | Func(func2) -> build_func overwrite_any func1 func2
          | _ -> raise Collect_Constraints_Error
        end
    | List(l_typ1) ->
        begin match typ2 with
          | Any | Unconst -> typ1
          | List(l_typ2) -> build_list overwrite_any l_typ1 l_typ2
          | _ -> raise Collect_Constraints_Error
        end
    | _  ->
        if typ1 = typ2 || typ2 = Unconst || typ2 = Any then typ1
        else raise Collect_Constraints_Error

  (* Collects possible constraints and returns type that is as constrained as
   * possible. Any remains. *)
  and keep_any typ1 typ2 = match typ1 with
    | Any -> Any
    | Unconst -> typ2
    | Func(func1) ->
        begin match typ2 with
          | Any -> Any
          | Unconst -> typ1
          | Func(func2) -> build_func keep_any func1 func2
```

```
          | _ -> raise Collect_Constraints_Error
        end
    | List(l_typ1) ->
        begin match typ2 with
          | Any -> Any
          | Unconst -> typ1
          | List(l_typ2) -> build_list keep_any l_typ1 l_typ2
          | _ -> raise Collect_Constraints_Error
        end
    | _  ->
        if typ2 = Any then Any
        else if typ1 = typ2 || typ2 = Unconst then typ1
        else raise Collect_Constraints_Error

  in overwrite_any typ1 typ2, keep_any typ1 typ2

(* Returns true if has Unconst, otherwise false *)
and has_unconst = function
  | Unconst -> true
  | List(typ) -> has_unconst typ
  | Func(func) ->
    let is_param_unconst = List.map has_unconst func.param_types and
      is_ret_unconst = has_unconst func.return_type in
    List.mem true is_param_unconst || is_ret_unconst
  | _ -> false

(* Turns Unconst types to Any *)
and unconst_to_any = function
  | Unconst -> Any
  | List(typ) -> let typ' = unconst_to_any typ in List(typ')
  | Func(func) ->
      let param_types' = List.map unconst_to_any func.param_types and
        return_type' = unconst_to_any func.return_type in
      Func({ param_types = param_types'; return_type = return_type' })
  | _ as typ -> typ

(* Turns Any to Unconst - DOES NOT AFFECT FUNCTIONS *)
and any_to_unconst = function
  | Any -> Unconst
  | List(typ) -> let typ' = any_to_unconst typ in List(typ')
  | _ as typ -> typ

(* Returns true if has Any, otherwise false - DOES NOT DEAL WITH FUNCTIONS *)
and has_any = function
  | Any -> true
  | List(typ) -> has_any typ
  | _ -> false

(* Returns true if Num or Unconst, otherwise false *)
and is_num = function
  | Num | Unconst -> true
  | _ -> false
```

```
and is_list_of_num = function
  | List(Num) | List(Unconst) | Unconst -> true
  | _ -> false

(* Check list elements against constraint type, constrain if possible *)
and constrain_list_elems env acc const = function
  | [] -> env, Sast.Expr(Sast.Ldecl(List.rev acc), List(const))
  | (Sast.Expr(_, typ) as ew) :: tl ->
      let _, const_w_any = try collect_constraints typ const
        with
          | Collect_Constraints_Error -> list_error (List(const)) typ
          | _ as e -> raise e in
      (* MIGHT NEED TO CHECK IF UNCONST *)
      let env', ew' = constrain_ew env ew const_w_any in
      constrain_list_elems env' (ew' :: acc) const tl

(*********************************************
 * Semantic checking and tree SAST construction
 *********************************************)

(* Branching point *)
and check_expr env = function
  | Ast.Num_lit(x) -> env, Sast.Expr(Sast.Num_lit(x), Num)
  | Ast.String_lit(s) -> env, Sast.Expr(Sast.String_lit(s), String)
  | Ast.Bool_lit(b) -> env, Sast.Expr(Sast.Bool_lit(b), Bool)
  | Ast.Void_lit -> env, Sast.Expr(Sast.Void_lit, Void)
  | Ast.Id(id) -> check_id env id
  | Ast.Unop(op, e) -> check_unop env op e
  | Ast.Binop(e1, op, e2) -> check_binop env e1 op e2
  | Ast.Call(id, args) -> check_func_call env id args
  | Ast.Assign(id, e) -> check_assign env id e
  | Ast.LDecl(l) -> check_list env l
  | Ast.Fdecl(f) -> check_fdecl env "_anon" f
  | Ast.Dist(d) -> check_dist env d
  | Ast.Discr_dist(d) -> check_discr_dist env d
  | Ast.Cake(fdecl, args) -> check_cake env fdecl args
  | Ast.If(i, t, e) -> check_if env i t e

(* Find string key 'id' in the environment if it exists *)
and check_id env id =
  let var =
    if VarMap.mem id env.scope then VarMap.find id env.scope else
    if VarMap.mem id env.params then VarMap.find id env.params else
    var_error id in
  env, Sast.Expr(Sast.Id(var.name), var.s_type)

(* Unary operators *)
and check_unop env op e =
  let env', ew = check_expr env e in
  let Sast.Expr(_, typ) = ew in
  match op with
    | Not -> begin match typ with
      | Bool -> env', Sast.Expr(Sast.Unop(op, ew), Bool)
```

```
      (* Attempt to constrain variable type of ew to Bool *)
      | Unconst -> let env', ew' = constrain_ew env' ew Bool in
          env', Sast.Expr(Sast.Unop(op, ew'), Bool)
      | _ as t -> unop_error op t
    end
    | Sub -> begin match typ with
      | Num -> env', Sast.Expr(Sast.Unop(op, ew), Num)
      (* Attempt to constrain variable type of ew to Num *)
      | Unconst -> let env', ew' = constrain_ew env' ew Num in
          env', Sast.Expr(Sast.Unop(op, ew'), Num)
      | _ as t -> unop_error op t
    end

(* Binary operators *)
and check_binop env e1 op e2 =
  if is_sugar op then (* Check if binop is sugar. If so unsugar. *)
    let func_name, e1', e2' = match op with
        | Cons -> ".cons", e1, e2
        | D_Plus -> ".dist_add", e1, e2
        | D_Times -> ".dist_mult", e1, e2
        | D_Shift -> ".dist_shift", e2, e1
        | D_Stretch -> ".dist_stretch", e2, e1
        | D_Power -> ".dist_exp", e2, e1
        | D_Sample -> ".dist_sample", e2, e1
        | _ -> dead_code_path_error "check_binop" in

    (* Unsugar expression and refeed it to Analyzer *)
    let unsugared = Ast.Call(Ast.Id(func_name), [e1'; e2']) in
    check_expr env unsugared

  else (* binop is not sugar. Check, constrain, and proceed. *)
    let env', ew1 = check_expr env e1 in
    let Sast.Expr(_, typ1) = ew1 in
    let env', ew2 = check_expr env' e2 in
    let Sast.Expr(_, typ2) = ew2 in
    match op with
      (* Numeric operations *)
      | Add | Sub | Mult | Div | Mod | Pow | Less | Leq | Greater | Geq ->
        if is_num typ1 && is_num typ2 then
          let result_type = match op with
            | Add | Sub | Mult | Div | Mod | Pow -> Num
            | Less | Leq | Greater | Geq -> Bool
            | _ -> dead_code_path_error "check_binop" in
          (* Constrain variable types to Num if neccessary *)
          let env', ew1' = constrain_ew env' ew1 Num in
          let env', ew2' = constrain_ew env' ew2 Num in
          env', Sast.Expr(Sast.Binop(ew1', op, ew2'), result_type)
        else binop_error typ1 op typ2

      (* Equality operations - overloaded, no constraining can be done, can take
       * any type *)
      | Eq | Neq -> env', Sast.Expr(Sast.Binop(ew1, op, ew2), Bool)
```

```
      (* Boolean operations *)
      | Or | And ->
        let is_bool = function
          | Bool | Unconst -> true
          | _ -> false in
        if is_bool typ1 && is_bool typ2 then
          (* Constrain variable types to Bool if necessary *)
          let env', ew1' = constrain_ew env' ew1 Bool in
          let env', ew2' = constrain_ew env' ew2 Bool in
          env', Sast.Expr(Sast.Binop(ew1, op, ew2), Bool)
        else binop_error typ1 op typ2

      | _ -> dead_code_path_error "check_binop"

(* Function calling *)
and check_func_call env id args =
  let env', ew = check_expr env id in
  let Sast.Expr(id', typ) = ew in
  let env', ew', f = match typ with
    | Sast.Func(f) -> env', ew, f
    | Unconst ->
        let f = {
          param_types = List.map (fun _ -> Unconst) args;
          return_type = Unconst;
        } in
        let env', ew' = constrain_ew env' ew (Func(f)) in env', ew', f
    | _ -> fcall_nonfunc_error id' typ in
  let env', args = check_func_call_args env' id' f args in
  let env', ret_typ = check_func_call_ret env' id args f.return_type in
  let env', ew' = check_expr env' id in
  env', Sast.Expr(Sast.Call(ew', args), ret_typ)

and check_func_call_args env id f args =
  if List.length f.param_types <> List.length args then
    fcall_length_error id (List.length f.param_types) (List.length args) else
  let rec aux env acc acc_param_types param_types = function
    | [] -> env, List.rev acc, List.rev acc_param_types
    | e :: tl -> let env', ew = check_expr env e in
        let Sast.Expr(_, typ) = ew in
        let param_type = List.hd param_types in
        let constrained, constrained_w_any = try collect_constraints typ param_type
          with
            | Collect_Constraints_Error -> fcall_argtype_error id typ param_type
            | _ as e -> raise e in
        let env', ew' =
          (* NOTE: IF THINGS ARE ACTING WEIRD ERROR IS PROBABLY ON LINE BELOW *)
          if has_unconst typ && typ <> constrained then
            constrain_ew env ew constrained
          else env', ew in
        aux env' (ew' :: acc) (constrained_w_any :: acc_param_types)
          (List.tl param_types) tl in

  let env', args', param_types' = aux env [] [] f.param_types args in
```

```
  if param_types' <> f.param_types then
    let f_type = Func({ f with param_types = param_types'; }) in
    let env' = constrain_e env' id f_type in
    env', args'
  else env', args'

and check_func_call_ret env id args ret_default =
  let id' = match id with
    | Ast.Id(id') -> id'
    | _ -> dead_code_path_error "check_func_call_ret" in
  let is_builtin =
    if VarMap.mem id' env.scope then
      (VarMap.find id' env.scope).builtin
    else false in

  if not is_builtin then
    (* If ret_default is Any, make it Unconst. Else if ret_default is a List
     * or contains lists of Any, make it List of Unconsts or List of Lists of
     * Unconsts. *)
    let ret_default' = any_to_unconst ret_default in
    env, ret_default'

  else (* is builtin *)
    (* helper function - get return type on buitlin list_func call *)
    let get_ret_type_from_typ = function
      | List(t) -> any_to_unconst t
      | Func(func) ->
          begin match func.return_type with
            | List(t) -> any_to_unconst t
            | _ -> dead_code_path_error "check_func_call_ret"
          end
      | _ -> dead_code_path_error "check_func_call_ret" in

    match id' with
      | "head" -> let Sast.Expr(_, typ) = List.hd args in
          env, get_ret_type_from_typ typ
      | "tail" -> let Sast.Expr(_, typ) = List.hd args in env, typ
      | ".cons" ->
          let Sast.Expr(cons, c_typ) = List.hd args and
            Sast.Expr(l, l_typ) = List.hd (List.tl args) in
          let l_elem_typ = get_ret_type_from_typ l_typ in
          let const, _ = try collect_constraints c_typ l_elem_typ
            with
              | Collect_Constraints_Error -> list_cons_mismatch_error c_typ l_typ
              | _ as e -> raise e in
          (* constrain the element begin appended *)
          let env' = constrain_e env cons const in
          (* constrain the list's type *)
          let env' = constrain_e env' l (List(const)) in
          env', List(const)
      | _ -> env, ret_default
```

```
(* Assignment *)
and check_assign env id = function
  | Ast.Fdecl(f) -> check_fdecl env id f
  | _ as e -> let env', ew = check_expr env e in
      let Sast.Expr(_, typ) = ew in
      let env', name = add_to_scope env' id typ in
      env', Sast.Expr(Sast.Assign(name, ew), typ)

(* Lists *)
and check_list env l =
  (* Evaluate list elements, transforming to sast types and storing list type *)
  let rec process_list env acc const = function
    | [] -> env, List.rev acc, const
    | e :: tl -> let env', ew = check_expr env e in
      let Sast.Expr(_, typ) = ew in
      let const', _ = try collect_constraints const typ
        with
          | Collect_Constraints_Error -> list_error (List(const)) typ
          | _ as e -> raise e in
      process_list env' (ew :: acc) const' tl in

  let env', l', const = process_list env [] Unconst l in
  constrain_list_elems env' [] const l'

(* Function declaration *)
and check_fdecl env id f =
  (* Add function name to scope with unconstrained param types and return type
   * to allow recursion *)
  let f_type = Func({
    param_types = List.map (fun _ -> Unconst) f.params;
    return_type = Unconst;
  }) in

  (* Check if attempting to reassign an identifier belonging to the parent
   * function. If so, fail. If not, add the function to scope *)
  let env', name =
    if VarMap.mem id env.scope then
      let old_type = (VarMap.find id env.scope).s_type in
      match old_type with
        | _ -> add_to_scope env id f_type
    else add_to_scope env id f_type in

  (* Evaluate parameters, body, and return statement in local environment *)
  let func_env, param_ssids = check_fdecl_params env' f.params in
  let func_env, body = check_stmts func_env f.body in
  let func_env, _ = check_expr func_env f.return in

  (* Evaluate parameter and function types. Check if the types of the
   * parameters in the function type are the same as the types of the
   * paramter variables themselves. If not, throw an error. Constrain Unconst
   * paramters in both the function type and as variables where possible *)
  let rec check_params_type_mismatch env acc func_param_types = function
    | [] -> env, List.rev acc
```

```
    | ssid :: tl ->
        let var = VarMap.find (id_of_ssid ssid) func_env.params and
          func_param_type = List.hd func_param_types in

        (* Constrain Param to extent possible *)
        let constrained, constrained_w_any =
          try collect_constraints var.s_type func_param_type
          with
            | Collect_Constraints_Error ->
                type_mismatch_error id func_param_type var.s_type
            | _ as e -> raise e in

        (* Convert remaining Unconst to Any *)
        let constrained_w_any = unconst_to_any constrained_w_any in

        (* If constrained_param has constraints not present in var, then
         * constrain var's type *)
        let func_env' =
          if var.s_type <> constrained then
            constrain_e func_env (Sast.Id(ssid)) constrained
          else func_env in

        (* Recurse *)
        check_params_type_mismatch func_env' (constrained_w_any :: acc)
          (List.tl func_param_types) tl in

  let param_types, return_typ =
    let f_typ = (VarMap.find id func_env.scope).s_type in
    match f_typ with
      | Func(func) -> func.param_types, func.return_type
      | _ -> fdecl_reassign_error id f_typ in
  let func_env, param_types' =
    check_params_type_mismatch func_env [] param_types param_ssids in

  (* Re-evaluate function return type to see if it has been constrained above *)
  let func_env, return = check_expr func_env f.return in

  (* If return type is Unconst, convert to Any *)
  let Sast.Expr(_, ret_type) = return in
  let ret_type' = unconst_to_any ret_type in

  (* If return type constrained differently than in env, throw error *)
  let rec not_any_and_not_unconst = function
    (* Returns false if type is Any, Unconst, or List of these. Otherwise
     * returns true *)
    | Any | Unconst -> false
    | List(typ) -> not_any_and_not_unconst typ
    | _ -> true in
  if not_any_and_not_unconst return_typ && ret_type' <> return_typ then
    return_type_mismatch_error id return_typ ret_type'
  else

  (* Construct function declaration *)
```

```
  let fdecl = {
    f_name = name;
    params = param_ssids;
    body = body;
    return = return;
  } in

  (* Construct function type *)
  let f_type = Func({ param_types = param_types'; return_type = ret_type' }) in

  (* Update function type in environment and return expression wrapper *)
  let ew = Sast.Expr(Sast.Fdecl(fdecl), Unconst) in
  (* MIGHT NEED TO CHECK IF UNCONST *)
  constrain_ew env' ew f_type

and check_fdecl_params env param_list =
  let rec aux env acc = function
    | [] -> env, List.rev acc
    | param :: tl -> let env', name = add_to_params env param in
        aux env' (name :: acc) tl
  in aux env [] param_list

(* Caking *)
and check_cake env fdecl args =
  let env', fdecl_ew = check_expr env fdecl in
  let env', call_ew = check_func_call env' (Id("_anon")) args in
  let Sast.Expr(_, typ) = call_ew in
  env', Sast.Expr(Sast.Cake(fdecl_ew, call_ew), typ)

(* Conditionals *)
and check_if env i t e =
  let env', ew1 = check_expr env i in
  let Sast.Expr(_, typ1) = ew1 in
  let env', ew1' = match typ1 with
    | Unconst -> constrain_ew env' ew1 Bool
    | Bool -> env, ew1
    | _ as typ -> typ_mismatch Bool typ in
  let env', ew2 = check_expr env' t in
  let Sast.Expr(_, typ2) = ew2 in
  let env', ew3 = check_expr env' e in
  let Sast.Expr(_, typ3) = ew3 in
  let const, _ = try collect_constraints typ2 typ3
    with
      | Collect_Constraints_Error -> if_mismatch_error typ2 typ3
      | _ as e -> raise e in
  let env', ew2' = if has_unconst typ2 then constrain_ew env' ew2 const
    else env', ew2 in
  let env', ew3' = if has_unconst typ3 then constrain_ew env' ew3 const
    else env', ew3 in
  let ifdecl = {
      c_name = (get_ssid "_cond");
      cond = ew1';
      stmt_1 = ew2';
```

```
      stmt_2 = ew3';
  } in
  env', Sast.Expr(Sast.If(ifdecl), const)

(* Distrubutions *)
and check_dist env d =
  (* Dists must have function of the following type: *)
  let dfunc_type = Func({ param_types = [Num]; return_type = Num; }) in

  (* Check and constrain min/max if neccessary *)
  let env', ew1 = check_expr env d.min in
  let Sast.Expr(_, typ1) = ew1 in
  let env', ew2 = check_expr env d.max in
  let Sast.Expr(_, typ2) = ew2 in
  let env', ew1', ew2' =
    if is_num typ1 && is_num typ2 then
      let env', ew1' = constrain_ew env' ew1 Num in
      let env', ew2' = constrain_ew env' ew2 Num in
      env', ew1', ew2'
    else invalid_dist_min_max_error typ1 typ2 in

  (* Check and constrain distribution function *)
  let env', ew3 = check_expr env d.dist_func in
  let Sast.Expr(_, typ3) = ew3 in
  let const, _ = try collect_constraints typ3 dfunc_type
    with
      | Collect_Constraints_Error -> invalid_dist_func_type_error typ3 dfunc_type
      | _ as e -> raise e in
  let env', ew3' =
    if has_unconst typ3 then constrain_ew env' ew3 const else env', ew3 in

  (* Construct Dist expr_wrapper *)
  let dist = Sast.Expr(Sast.Dist({
    min = ew1'; max = ew2'; dist_func = ew3';
  }), Dist_t) in

  (* Return Dist expr_wrapper *)
  env', dist

and check_discr_dist env d =
  (* Check and constrain min/max if neccessary *)
  let env', ew1 = check_expr env d.vals in
  let Sast.Expr(_, typ1) = ew1 in
  let env', ew2 = check_expr env d.weights in
  let Sast.Expr(_, typ2) = ew2 in
  let env', ew1', ew2' =
    if is_list_of_num typ1 && is_list_of_num typ2 then
      let env', ew1' = constrain_ew env' ew1 (List(Num)) in
      let env', ew2' = constrain_ew env' ew2 (List(Num)) in
      env', ew1', ew2'
    else invalid_discr_dist_error typ1 typ2 in

  (* Construct Dist expr_wrapper *)
```

```
  let dist = Sast.Expr(Sast.Discr_dist({
    vals = ew1'; weights = ew2';
  }), Dist_t) in

  (* Return Dist expr_wrapper *)
  env', dist

(* Statements *)
and check_stmt env = function
  | Ast.Do(e) -> let env', ew = check_expr env e in env', Sast.Do(ew)

and check_stmts env stmt_list =
  let rec aux env acc = function
    | [] -> env, List.rev acc
    | stmt :: tl -> let env', e = check_stmt env stmt in
        aux env' (e :: acc) tl
  in aux env [] stmt_list

(* Program entry point *)
let check_ast ast =
  let _, sast = check_stmts root_env ast in sast
```

sast.mli

```
(*
 * COMS4115: Odds semantically checked abstract syntax tree
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

open Ast

(* Data types *)
type data_type =
  | Num
  | String
  | Bool
  | Void
  | List of data_type
  | Func of func
  | Any
  | Dist_t
  | Unconst

and func = {
  param_types: data_type list;
  return_type: data_type;
}
```

```
type var = {
  name: string;
  mutable s_type: data_type;
  builtin: bool;
}

(* Expressions *)
type expr_wrapper =
  | Expr of expr * data_type

and expr =
  | Num_lit of Ast.num
  | String_lit of string
  | Bool_lit of bool
  | Void_lit
  | Unop of Ast.unop * expr_wrapper
  | Binop of expr_wrapper * Ast.binop * expr_wrapper
  | Id of string
  | Assign of string * expr_wrapper
  | Call of expr_wrapper * expr_wrapper list
  | Dist of dist
  | Discr_dist of discr_dist
  | Ldecl of expr_wrapper list
  | Fdecl of fdecl
  | Cake of expr_wrapper * expr_wrapper
  | If of ifdecl

and dist = {
  min: expr_wrapper;        (* Distribution Minimum *)
  max: expr_wrapper;        (* Distribution Maximum *)
  dist_func: expr_wrapper;  (* Distribution Function *)
}

and discr_dist = {
  vals: expr_wrapper;        (* Distribution Values *)
  weights: expr_wrapper;     (* Distribution Weights *)
}

and fdecl = {
  f_name: string;         (* Function Name *)
  params: string list;    (* Parameters *)
  body: stmt list;        (* Function Body *)
  return: expr_wrapper;   (* Return *)
}

and ifdecl = {
  c_name: string;         (* Function Name *)
  cond: expr_wrapper;     (* If *)
  stmt_1: expr_wrapper;   (* Then *)
  stmt_2: expr_wrapper;   (* Else *)
}
```

```
(* Statements *)
and stmt =
  | Do of expr_wrapper    (* set foo = bar + 3 *)

(* Program entry point *)
type program = stmt list
```

pythonizer.ml

```
(*
 * COMS4115: Python AST Generator
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

open Sast
open Past

exception Python_Error of string

(* Strip private '.' prefix designation for ID if it exists *)
let private_to_normal id =
  if id.[0] = '.' then String.sub id 1 (String.length id - 1) else id

(* Expressions *)
let rec past_expr stmts = function
  | Sast.Num_lit(n) -> stmts, Past.Num_lit(n)
  | Sast.String_lit(s) -> stmts, Past.String_lit(s)
  | Sast.Bool_lit(b) -> stmts, Past.Bool_lit(b)
  | Sast.Void_lit -> stmts, Past.None_lit
  | Sast.Id(id) -> let id' = private_to_normal id in stmts, Past.Id(id')
  | Sast.Unop(op, we) -> let stmts', e = past_expr_unwrap stmts we in
      stmts', Past.Unop(op, e)
  | Sast.Binop(we1, op, we2) ->
      let stmts', e1 = past_expr_unwrap stmts we1 in
      let stmts', e2 = past_expr_unwrap stmts' we2 in
      stmts', Past.Binop(e1, op, e2)
  | Sast.Call(wid, wargs) ->
      let stmts', id = past_expr_unwrap stmts wid in
      let stmts', args = past_list stmts' wargs in
      stmts', Past.Call(id, args)
  | Sast.Ldecl(wl) ->
      let stmts', l = past_list stmts wl in stmts', Past.Ldecl(l)
  | Sast.Dist(d) -> past_dist stmts d
  | Sast.Discr_dist(d) -> past_discr_dist stmts d
  | Sast.Assign(id, we) -> let stmts', e = past_expr_unwrap stmts we in
      (Past.Assign(id, e) :: stmts'), Past.Id(id)
  | Sast.Fdecl(f) -> let stmts', def = past_fdecl stmts f in
```

```
      (Past.Def(def) :: stmts'), Past.Id(def.p_name)
  | Sast.Cake(wfdecl, wcall) -> let stmts', _ = past_expr_unwrap stmts wfdecl in
      past_expr_unwrap stmts' wcall
  | Sast.If(cond) -> mk_if_function stmts cond

and past_expr_unwrap stmts = function
  | Sast.Expr(e, _) -> past_expr stmts e

(* Distributions *)
and past_dist stmts d =
  let stmts1, min' = past_expr_unwrap stmts d.min in
  let stmts2, max' = past_expr_unwrap stmts1 d.max in
  let stmts3, dist_func' = past_expr_unwrap stmts2 d.dist_func in
  stmts3, Past.Call(Past.Id("make_dist"), [min' ; max' ; dist_func'])

and past_discr_dist stmts d =
  let stmts1, vals' = past_expr_unwrap stmts d.vals in
  let stmts2, weights' = past_expr_unwrap stmts1 d.weights in
  stmts2, Past.Call(Past.Id("make_discr_dist"), [vals' ; weights'])

(* Lists *)
and past_list stmts expr_list =
  let rec aux stmts acc = function
    | [] -> stmts, List.rev acc
    | we :: tl -> let stmts', e = past_expr_unwrap stmts we in
        aux stmts' (e :: acc) tl
  in aux stmts [] expr_list

(* Functions *)
and mk_if_function stmts cond =
    let stmts', i = past_expr_unwrap stmts cond.cond in
    let stmts', t = past_expr_unwrap stmts' cond.stmt_1 in
    let stmts', e = past_expr_unwrap stmts' cond.stmt_2 in
    let r1 = Past.Return(t) in
    let r2 = Past.Return(e) in
    let if_stmt = Past.If(i, r1, r2) in
    let f = {
      p_name = cond.c_name;
      p_params = [];
      p_body = [if_stmt];
    } in
    let stmts' = (Def(f) :: stmts') in
    stmts', Past.Call(Past.Id(f.p_name), [])

and past_fdecl stmts sast_f =
  let body = past_stmts sast_f.body in
  let body', e = past_expr_unwrap body sast_f.return in
  let r = Past.Return(e) in
  let body' = body' @ [r] in
  let f = {
    p_name = sast_f.f_name;
    p_params = sast_f.params;
    p_body = body';
```

```
  } in stmts, f

(* Statements *)
and past_stmt stmts = function
  | Sast.Do(we) -> let stmts', e = past_expr_unwrap stmts we in stmts', e

and past_stmts stmt_list =
  let rec aux acc = function
    | [] -> List.rev acc
    | stmt :: tl -> let stmts', s = past_stmt acc stmt in
        aux (Past.Stmt(s) :: stmts') tl
  in aux [] stmt_list

(* Program entry point *)
let generate_past sast = past_stmts sast
```

past.mli

```
(*
 * COMS4115: Odds python abstract syntax tree
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

open Ast

(* Expressions *)
type expr =
  | Num_lit of Ast.num
  | String_lit of string
  | None_lit
  | Bool_lit of bool
  | Unop of Ast.unop * expr
  | Binop of expr * Ast.binop * expr
  | Id of string
  | Call of expr * expr list
  | Ldecl of expr list

(* Function Declarations *)
and fdecl = {
  p_name: string;          (* Function Name *)
  p_params: string list;   (* Parameters *)
  p_body: stmt list;       (* Function Body *)
}

(* Statements *)
and stmt =
  | Return of expr
```

```
  | Def of fdecl
  | If of expr * stmt * stmt
  | Assign of string * expr
  | Stmt of expr

(* Program entry point *)
type program = stmt list
```

generator.ml

```
(*
 * COMS4115: Odds Python code generator
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

open Ast
open Past
open Printf

exception Python_Error of string

(* Indentation *)
let indent_of_num indent = String.make (4 * indent) ' '

(* Unary operators *)
let txt_of_unop = function
  | Not -> "not "
  | Sub -> "-"

(* Binary operators *)
let txt_of_num = function
  | Num_int(i) -> string_of_int i
  | Num_float(f) -> string_of_float f

let txt_of_binop = function
  (* Arithmetic *)
  | Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Mod -> "%"
  | Pow -> "**"
  (* Boolean *)
  | Or -> "or"
  | And -> "and"
  | Eq -> "=="
  | Neq -> "!="
```

```
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | _ -> raise (Python_Error "Unexpected binary operator")

(* Conditionals *)
let txt_of_cond indent i t e = sprintf "%sif %s:\n%s\n%s"
  (indent_of_num indent) i t e

(* Expressions *)
let rec txt_of_expr = function
  | Num_lit(n) -> txt_of_num n
  | String_lit(s) -> sprintf "\"%s\"" s
  | Bool_lit(b) -> String.capitalize (string_of_bool(b))
  | None_lit -> "None"
  | Id(id) -> id
  | Unop(op, e) -> sprintf "(%s%s)" (txt_of_unop op) (txt_of_expr e)
  | Binop(e1, op, e2) -> sprintf "(%s %s %s)"
      (txt_of_expr e1) (txt_of_binop op) (txt_of_expr e2)
  | Call(id, args) -> txt_of_call id args
  | Ldecl(l) -> sprintf "[%s]" (txt_of_list l)

(* Function calls *)
and txt_of_call id args = match id with
  | Id("head") -> sprintf "%s[0]" (txt_of_expr (List.hd args))
  | Id("tail") -> sprintf "%s[1:]" (txt_of_expr (List.hd args))
  | Id("cons") ->
      let prepend = txt_of_expr (List.hd args) and
        list_txt = txt_of_expr (List.hd (List.tl args)) in
      sprintf "([%s] + %s)" prepend list_txt
  | _ -> sprintf "%s(%s)" (txt_of_expr id) (txt_of_list args)

(* Lists *)
and txt_of_list = function
  | [] -> ""
  | [x] -> txt_of_expr x
  | _ as l -> let strs = List.map (fun x -> txt_of_expr x) l in
    String.concat ", " strs

(* Functions *)
and txt_of_fdecl indent f =
    let params = String.concat ", " f.p_params in
    let body = txt_of_stmts (indent + 1) f.p_body in
    sprintf "%sdef %s(%s):\n%s"
      (indent_of_num indent)
      f.p_name
      params
      body

(* Statements *)
and txt_of_stmt indent = function
  | Assign(id, e) -> sprintf "%s%s = %s"
```

```
      (indent_of_num indent) id (txt_of_expr e)
  | Def(f) -> txt_of_fdecl indent f
  | Return(e) -> sprintf "%sreturn %s" (indent_of_num indent) (txt_of_expr e)
  | If(i, t, e) ->
      let i' = txt_of_expr i
        and t' = txt_of_stmt (indent + 1) t
        and e' = txt_of_stmt indent e in
      txt_of_cond indent i' t' e'
  | Stmt(e) -> sprintf "%s%s" (indent_of_num indent) (txt_of_expr e)

and txt_of_stmts indent stmt_list =
  let rec aux indent acc = function
    | [] -> String.concat "\n" (List.rev acc)
    | stmt :: tl -> aux indent ((txt_of_stmt indent stmt) :: acc) tl
  in aux indent [] stmt_list

(* Code generation entry point *)
let gen_program past = txt_of_stmts 0 past
```

printer.ml

```
(*
 * COMS4115: Odds pretty printer for semantically checked abstract syntax tree
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

open Ast
open Sast
open Analyzer
open Printf

exception Printer_Error of string

(* Utility Functions *)
let tabsize = 2
let tabs = ref 0
let tab_str () = String.make (!tabs * tabsize) ' '

let str_of_colored_type typ =
  sprintf "\x1b[34m%s\x1b[0m" (Analyzer.str_of_type typ)

(* Stringerizer *)
let rec str_of_expr_wrapper = function
  | Sast.Expr(Num_lit(n), _) ->
      begin match n with
        | Ast.Num_int(i) -> string_of_int i
        | Ast.Num_float(f) -> string_of_float f
```

```
      end
  | Sast.Expr((String_lit(s)), _) -> sprintf "\"%s\"" s
  | Sast.Expr(Bool_lit(b), _) -> string_of_bool b
  | Sast.Expr(Void_lit, _) -> "void"
  | Sast.Expr(Unop(op, we), _) ->
      let op_str = Analyzer.str_of_unop op and
        we_str = str_of_expr_wrapper we in
      sprintf "%s%s" op_str we_str
  | Sast.Expr(Binop(we1, op, we2), _) ->
      let we1_str = str_of_expr_wrapper we1 and
        op_str = Analyzer.str_of_binop op and
        we2_str = str_of_expr_wrapper we2 in
      sprintf "%s %s %s" we1_str op_str we2_str
  | Sast.Expr(Id(id), typ) -> let typ_str = str_of_colored_type typ in
      sprintf "%s %s" typ_str id
  | Sast.Expr(Assign(id, we), typ) ->
      let we_str = str_of_expr_wrapper we and
        typ_str = str_of_colored_type typ in
      sprintf "%s %s = %s" typ_str id we_str
  | Sast.Expr(Call(we, we_list), _) ->
      let func_name = str_of_expr_wrapper we and
        args_txt = str_of_expr_wrapper_list we_list in
      sprintf "%s(%s)" func_name args_txt
  | Sast.Expr(Ldecl(we_list), _) ->
      let l_txt = str_of_expr_wrapper_list we_list in
      sprintf "[%s]" l_txt
  | Sast.Expr(Dist(dist), _) ->
      let min_txt = str_of_expr_wrapper dist.min and
        max_txt = str_of_expr_wrapper dist.max and
        func_txt = str_of_expr_wrapper dist.dist_func in
      sprintf "<%s, %s>|%s|" min_txt max_txt func_txt
  | Sast.Expr(Discr_dist(dist), _) ->
      let values = str_of_expr_wrapper dist.vals and
        weights = str_of_expr_wrapper dist.weights in
      sprintf "<%s, %s>" values weights

  | Sast.Expr(Fdecl(fdecl), typ) -> str_of_fdecl fdecl typ
  | Sast.Expr(Cake(fdecl_ew, call_ew), _) -> str_of_cake fdecl_ew call_ew
  | Sast.Expr(If(cond), _) -> str_of_cond cond

and str_of_expr_wrapper_list l =
  String.concat ", " (List.map str_of_expr_wrapper l)

and str_of_fdecl fdecl typ =
  let str_of_param_and_type typ param =
    sprintf "%s %s" (str_of_colored_type typ) param in

  let func = match typ with
    | Func(func) -> func
    | _ -> raise (Printer_Error "Function has non-function type") in

  tabs := !tabs + 1;
  let params_and_types = List.map2 str_of_param_and_type func.param_types
```

```
fdecl.params and
    return_type = str_of_colored_type func.return_type in

  let decl_txt = sprintf "%s => %s" (String.concat ", " params_and_types)
return_type and
    body_txt = str_of_stmts fdecl.body and
    return_txt = str_of_expr_wrapper fdecl.return in

  let f_str =
    let is_body = String.length body_txt > 0 in
    sprintf "%s(%s) ->%s%sreturn %s" fdecl.f_name decl_txt
      (if is_body then "\n" ^ body_txt ^ "\n" else "")
      (if is_body then tab_str () else " ")
      (if is_body then return_txt ^ "\n" else return_txt) in
  tabs := !tabs - 1; f_str

(* Currently not using *)
and str_of_cake fdecl_ew call_ew =
  tabs := !tabs + 1;

  let fdecl, f_typ = match fdecl_ew with
    | Sast.Expr(Sast.Fdecl(fdecl), f_typ) -> fdecl, f_typ
    | _ -> raise (Printer_Error "Dead Code Path") in

  let fdecl_txt = str_of_fdecl fdecl f_typ and
    call_txt = str_of_expr_wrapper call_ew in
  let c_str = sprintf "{%s}%s" fdecl_txt call_txt in
  tabs := !tabs - 1; c_str

and str_of_cond cond =
  tabs := !tabs + 1;
  let tabins = (tab_str ()) ^ (String.make tabsize ' ') in

  let cond_str = sprintf "\n%sif %s then\n%s%s\n%selse\n%s%s"
    (tab_str ()) (str_of_expr_wrapper cond.cond) tabins
    (str_of_expr_wrapper cond.stmt_1) (tab_str ()) tabins
    (str_of_expr_wrapper cond.stmt_2) in
  tabs := !tabs - 1; cond_str

and str_of_stmt = function
  | Sast.Do(wrapped_expr) ->
      sprintf "%sdo %s" (tab_str ()) (str_of_expr_wrapper wrapped_expr)

and str_of_stmts sast =
  let rec aux acc = function
    | [] -> String.concat "\n" (List.rev acc)
    | hd :: tl -> aux (str_of_stmt hd :: acc) tl
  in aux [] sast

let print_sast sast =
    let sast_str = str_of_stmts sast in
    print_endline sast_str
```

core.py

```
"""
  COMS4115: A compiled Odds program.

  Authors:
    - Alex Kalicki
    - Alexandra Medway
    - Daniel Echikson
    - Lilly Wang
"""
from __future__ import print_function
import math
import random
import sys

# Odds constants
EUL = math.e
PI = math.pi

INDEX_STEP = 1000
DIST_LENGTH = 10000
SAMPLE_STEP = 100

PLOT = False

def exception(s):
    """Write exception s to stderr and exit program"""
    sys.stderr.write("%s\n" % s)
    exit(1)

def print(*args, **kwargs):
    """Plot distributions for long lists and call normal print() function,
    but return argument that was passed"""
    if type(args[0]) is list and len(args[0]) >= DIST_LENGTH:
        print_dist(args[0])
        return str(args[0])
    __builtins__.print(*args, **kwargs)
    return str(args[0])

def print_dist(dist):
    """Opens a new figure (window) for each distribution it prints, removes
    the y-axis labels, and does not show them all until the end"""
    import matplotlib.pyplot as plt
    global PLOT
    PLOT = True
    plt.figure()
    plt.hist(dist, bins=20, normed=True)
    ax = plt.gca()
    ax.axes.get_yaxis().set_visible(False)

def make_dist(start, end, f):
    """Return a list generated from dist<min, max> | f"""
```

```
    if end <= start:
        exception("dist_make: start cannot be greater than end")
    step = (end - start) * 1.0 / INDEX_STEP
    indices = [ start + step * x for x in range(INDEX_STEP) ]

    cum_sum = 0.0
    cum_weights = []
    for x in indices:
        cum_sum += abs(f(x))
        cum_weights.append(cum_sum)
    rands = sorted([ random.uniform(0, cum_sum) for x in range(DIST_LENGTH) ])

    cum_i = 0
    rand_i = 0
    dist_list = []
    while rand_i < len(rands):
        if rands[rand_i] < cum_weights[cum_i]:
            dist_list.append(indices[cum_i])
            rand_i = rand_i + 1
        else:
            cum_i = cum_i + 1
    return dist_list


def dist_add(d1, d2):
    """Return the sum of two distributions, adding each combination"""
    s1 = d1[random.randint(0, SAMPLE_STEP - 1)::SAMPLE_STEP]
    s2 = d2[random.randint(0, SAMPLE_STEP - 1)::SAMPLE_STEP]
    return sorted([ x + y for x in s1 for y in s2 ])

def dist_mult(d1, d2):
    """Return the product of two distributions, multiplying each combination"""
    s1 = d1[random.randint(0, SAMPLE_STEP - 1)::SAMPLE_STEP]
    s2 = d2[random.randint(0, SAMPLE_STEP - 1)::SAMPLE_STEP]
    return sorted([ x * y for x in s1 for y in s2 ])

def make_discr_dist(vals, weights):
    """Return a list generated from dist<vals, weights>"""
    if len(vals) != len(weights):
        exception("dist_make: discrete dist with different sized lists")

    cum_weights = [sum(weights[:i+1]) for i in xrange(len(weights))]
    rands = sorted([ random.uniform(0, max(cum_weights)) for x in
range(DIST_LENGTH) ])

    cum_i = 0
    rand_i = 0
    dist_list = []
    while rand_i < len(rands):
        if rands[rand_i] < cum_weights[cum_i]:
            dist_list.append(vals[cum_i])
            rand_i = rand_i + 1
        else:
```

```
            cum_i = cum_i + 1
    return dist_list

def dist_shift(n, d):
    """Shift each element in distribution d by n"""
    return [ x + n for x in d ]

def dist_stretch(n, d):
    """Stretch distribution d, multiplying each element by n"""
    return [ x * n for x in d ]

def dist_exp(n, d):
    """Exponentiate distribution d, raising each element to power n"""
    return [ x ** n for x in d ]

def dist_sample(n, d):
    """Return a random sample of n elements in distribution d"""
    return sorted([ random.randint(0, DIST_LENGTH - 1) for x in range(n) ])

def P(n, d):
    """Return the probability that the distribution is less than
    the inputed value
    """
    return len([i for i in d if i < n]) * 1.0 / DIST_LENGTH

def E(d):
    """ Expected value of the distribution """
    return sum(d) * 1.0 / DIST_LENGTH

"""
END ODDS CORE LIBRARY
BEGIN USER CODE
"""
```

dist.ods

```
/*
 * COMS4115: ODDS Dist Standard Library
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 */

do uniform = (x) -> return 1

do normal = (x) ->
    do coef = 1 / (2 * PI) ** 0.5
    do exp = -1 * x ** 2 / 2
    return coef * EUL ** exp
```

list.ods

```
/*
 * COMS4115: ODDS List Standard Library
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 */

/* Return true if list is empty, false otherwise */
do list_empty = (l) -> return len(l) == 0

/*
 * Create a list with of n values initialized to v. Throw exception with
 * message "list_make: n can not be negative" if n is negative
 */
do list_make = (n, v) ->
    do if n < 0 then exception("list_make: n can not be negtive") else void
    do aux = (acc, n) ->
        return if n == 0 then acc else aux(v :: acc, n - 1)
    return aux([], n)

/*
 * Return the nth element in the list, wiith the head at position 0. Throw
 * exception with message "list_get: List index out of range" if list too short,
 * message "list_nth: n can not be negative" if n is negative
 */
do list_get = (n, l) ->
    do
        if n < 0 then exception("list_get: n can not be negative") else
        if len(l) <= n then exception("list_get: List index out of range")
        else void
    return
        if n == 0 then head(l)
        else list_get(n - 1, tail(l))

/*
 * Apply a function to a partial result and an element of the list to produce
 * the next partial result.
 */
do list_fold = (f, acc, l) ->
    return
        if list_empty(l) then acc
        else list_fold(f, f(acc, head(l)), tail(l))

/* Return list l with elements in reverse order */
do list_rev = (l) ->
    do aux = (acc, l) ->
        return
            if list_empty(l) then acc
```

```
            else aux(head(l) :: acc, tail(l))
    return aux([], l)

/* Concatenate lists a and b and return the result */
do list_concat = (a, b) ->
    do aux = (acc, l) ->
        return
            if list_empty(l) then list_rev(acc)
            else aux(head(l) :: acc, tail(l))
    return aux(list_rev(a), b)

/* Returns a new list of l's elements with function f applied */
do list_map = (f, l) ->
    do aux = (acc, l) ->
        return
            if list_empty(l) then list_rev(acc)
            else (() ->
                do applied = f(head(l))
                return aux(applied :: acc, tail(l))
            )()
    return aux([], l)

/* Same as list_map, but applied f and discards the return function */
do list_iter = (f, l) ->
    do list_map(f, l)
    return void

/*
 * Return list with the specified value inserted before the specified index.
 * Throw exception with message "list_insert: List index out of range" if list
 * too short to insert at given index, "list_insert: i can not be negative" if
 * negative index supplied.
 */
do list_insert = (v, i, l) ->
    do
        if i < 0 then exception("list_insert: i can not be negative") else
        if len(l) < i then exception("list_insert: List index out of range")
        else void
    do aux = (acc, i, l) ->
        return
            if i == 0 then list_concat(list_rev(v :: acc), l)
            else aux(head(l) :: acc, i - 1, tail(l))
    return aux([], i, l)

/*
 * Return modified with the specified index removed. Throw exception with
 * message "list_remove: List index out of range" if list too short to remove at
 * given index, "list_remove: i can not be negative" if negative index supplied.
 */
do list_remove = (i, l) ->
    do
        if i < 0 then exception("list_remove: i can not be negative") else
        if len(l) <= i then exception("list_remove: List index out of range")
```

```
        else void
    do aux = (acc, i, l) ->
        return
            if i == 0 then list_concat(list_rev(acc), tail(l))
            else aux(head(l) :: acc, i - 1, tail(l))
    return aux([], i, l)
```

utils.ml

```
(*
 * COMS4115: Odds Utility File
 *
 * Authors:
 *  - Alex Kalicki
 *  - Alexandra Medway
 *  - Daniel Echikson
 *  - Lilly Wang
 *)

(* Return a string representation of file 'file' *)
let str_of_file file =
  let ic = open_in file in
  let try_read () =
    try Some(input_line ic) with End_of_file -> None in
  let rec aux acc = match try_read () with
    | None -> close_in ic; String.concat "\n" (List.rev acc)
    | Some(s) -> aux (s :: acc) in
  aux []

let conclude_program () = "if PLOT:\n\timport matplotlib.pyplot as
plt\n\tplt.show()"
```

12.2 Test Code

We wrote tests as we programmed, so the group was collectively responsible for the testing code.

12.2.1 Scanner Tests

scanner_test.sh

```
#!/bin/bash

NC='\033[0m'
CYAN='\033[0;36m'
GREEN='\033[0;32m'
RED='\033[0;31m'

INPUT_FILES="scanner/*.in"
```

```
printf "${CYAN}Running scanner tests...\n${NC}"

for input_file in $INPUT_FILES; do
    output_file=${input_file/.in/.out}
    scanner/tokenize < $input_file | cmp -s $output_file -
    if [ "$?" -eq 0 ]; then
       printf "%-65s ${GREEN}SUCCESS\n${NC}" "  - checking $input_file..."
    else
       printf "%-65s ${RED}ERROR\n${NC}" "  - checking $input_file..." 1>&2
       exit 1
    fi
done

exit 0
```

tokenize.ml

```
open Parser
open Ast

type num =
  | Num_int of int
  | Num_float of float

let stringify = function
  (* Punctuation *)
  | LPAREN -> "LPAREN"  | RPAREN -> "RPAREN"
  | LCAR -> "LCAR"      | RCAR -> "RCAR"
  | LBRACE -> "LBRACE"  | RBRACE -> "RBRACE"
  | COMMA -> "COMMA"    | VBAR -> "VBAR"
  | DDELIM -> "DDELIM"  | DISC -> "DISC"

  (* Dist Operators *)
  | DPLUS -> "DPLUS"    | DTIMES -> "DTIMES"
  | DPOWER -> "DPOWER"  | DSHIFT -> "DSHIFT"
  | DSTRETCH -> "DSTRETCH"

  (* Arithmetic Operators *)
  | PLUS -> "PLUS"     | MINUS -> "MINUS"
  | TIMES -> "TIMES"   | DIVIDE -> "DIVIDE"
  | MOD -> "MOD"       | POWER -> "POWER"

  (* Relational Operators *)
  | EQ -> "EQ"    | NEQ -> "NEQ"
  | LEQ -> "LEQ"  | GEQ -> "GEQ"

  (* List Operators *)
  | CONS -> "CONS"

  (* Logical Operators & Keywords *)
  | AND -> "AND"   | OR -> "OR"
```

```
  | NOT -> "NOT"

  (* Assignment Operator *)
  | ASN -> "ASN"

  (* Conditional Operators *)
  | IF -> "IF"    | THEN -> "THEN"
  | ELSE -> "ELSE"

  (* Declarative Keywords *)
  | DO -> "DO"

  (* Function Symbols & Keywords *)
  | FDELIM -> "FDELIM"
  | RETURN -> "RETURN"
  | CAKE -> "CAKE"

  (* End-of-File *)
  | EOF -> "EOF"

  (* Identifiers *)
  | ID(string) -> "ID"

  (* Literals *)
  | NUM_LITERAL(num) -> "NUM_LITERAL"
  | STRING_LITERAL(string) -> "STRING_LITERAL"
  | BOOL_LITERAL(bool) -> "BOOL_LITERAL"
  | VOID_LITERAL -> "VOID_LITERAL"

let _ =
  let lexbuf = Lexing.from_channel stdin in
  let rec print_tokens = function
    | EOF -> " "
    | token ->
      print_endline (stringify token);
      print_tokens (Scanner.token lexbuf) in
  print_tokens (Scanner.token lexbuf)
```

_arithmetic.in

```
+ - * / % **
```

_arithmetic.out

```
PLUS
MINUS
TIMES
DIVIDE
MOD
POWER
```

_assignment.in

```
=
```

_assignment.out

```
ASN
```

_comment.in

```
/*
 * I am writing a comment
 * "Hello"
 * None of this should be tokenized.
 * set a = 4
 */
```

_comment.out

```

```

_conditinal.in

```
if then else
```

_conditional.out

```
IF
THEN
ELSE
```

_declarative.in

```
do
```

_declarative.out

```
DO
```

_dist.in

```
<+> <*> |** |* |+
```

_dist.out

```
DPLUS
DTIMES
DPOWER
DSTRETCH
DSHIFT
```

_evenfunc.in

```
do iseven = (num) ->
  return if num % 2 == 0 then true else "false"
```

_evenfunc.out

```
DO
ID
ASN
LPAREN
ID
FDELIM
RETURN
IF
ID
MOD
NUM_LITERAL
EQ
NUM_LITERAL
THEN
BOOL_LITERAL
ELSE
STRING_LITERAL
```

_function.in

```
) -> return
) (
```

_function.out

```
FDELIM
RETURN
CAKE
```

_identifier.in

```
Aaaa
aa
a_a
a92
```

_identifier.out

```
ID
ID
ID
ID
```

_literal.in

```
"hello"
34.59
true
void
"\"x"
2
```

_literal.out

```
STRING_LITERAL
NUM_LITERAL
BOOL_LITERAL
VOID_LITERAL
STRING_LITERAL
NUM_LITERAL
```

_logical.in

```
&& || !
```

_logical.out

```
AND
OR
NOT
```

_mixed_arithmetic.in

```
10 + 10.34 ** 0.5 / 10-20.0
```

_mixed_arithmetic.out

```
NUM_LITERAL
PLUS
NUM_LITERAL
POWER
NUM_LITERAL
DIVIDE
NUM_LITERAL
MINUS
NUM_LITERAL
```

_punctuation.in

```
() <> [] , |
> |
```

_punctuation.out

```
LPAREN
RPAREN
LCAR
RCAR
LBRACE
RBRACE
COMMA
VBAR
DDELIM
```

_relational.in

```
== != < <= > >=
```

_relational.out

```
EQ
NEQ
LCAR
LEQ
RCAR
GEQ
```

12.2.1 Parser Tests

parser_test.sh

```
#!/bin/bash

NC='\033[0m'
CYAN='\033[0;36m'
GREEN='\033[0;32m'
RED='\033[0;31m'

INPUT_FILES="parser/*.in"
printf "${CYAN}Running parser tests...\n${NC}"

for input_file in $INPUT_FILES; do
    output_file=${input_file/.in/.out}
    input=$(parser/parserize < $input_file | tr -d "[:space:]")
    output=$(tr -d "[:space:]" < $output_file);
    if [[ "$input" == "$output" ]]; then
        printf "%-65s ${GREEN}SUCCESS\n${NC}" "  - checking $input_file..."
    else
        printf "%-65s ${RED}ERROR\n${NC}" "  - checking $input_file..." 1>&2
        exit 1
    fi
done
```

```
exit 0
```

_parserize.ml

```
open Ast
open Printf

(* Unary operators *)
let txt_of_unop = function
  | Not -> "Not"
  | Sub -> "Sub"

(* Binary operators *)
let txt_of_binop = function
  (* Dist *)
  | D_Plus -> "D_Plus"
  | D_Times -> "D_Times"
  | D_Shift -> "D_Shift"
  | D_Stretch -> "D_Stretch"
  | D_Power -> "D_Power"
  | D_Sample -> "D_Sample"
  (* Arithmetic *)
  | Add -> "Add"
  | Sub -> "Sub"
  | Mult -> "Mult"
  | Div -> "Div"
  | Mod -> "Mod"
  | Pow -> "Pow"
  (* Boolean *)
  | Or -> "Or"
  | And -> "And"
  | Eq -> "Eq"
  | Neq -> "Neq"
  | Less -> "Less"
  | Leq -> "Leq"
  | Greater -> "Greater"
  | Geq -> "Geq"
  | Cons -> "::"

(* Expressions *)
let txt_of_num = function
  | Num_int(x) -> string_of_int x
  | Num_float(x) -> string_of_float x

let rec txt_of_expr = function
  | Num_lit(x) -> sprintf "Num_lit(%s)" (txt_of_num x)
  | String_lit(x) -> sprintf "String_lit(%s)" x
  | Bool_lit(x) -> sprintf "Bool_lit(%s)" (string_of_bool x)
  | Void_lit -> "Void_lit"
  | Id(x) -> sprintf "Id(%s)" x
  | Unop(op, e) -> sprintf "Unop(%s, %s)" (txt_of_unop op) (txt_of_expr e)
```

```
  | Binop(e1, op, e2) -> sprintf "Binop(%s, %s, %s)"
      (txt_of_expr e1) (txt_of_binop op) (txt_of_expr e2)
  | Call(f, args) -> sprintf "Call(%s, [%s])"
      (txt_of_expr f) (txt_of_list args)
  | Assign(x, e) -> sprintf "Assign(%s, %s)" x (txt_of_expr e)
  | LDecl(l) -> sprintf "LDecl([%s])" (txt_of_list l)
  | Dist(d) -> txt_of_dist d
  | Discr_dist(d) -> txt_of_discr_dist d
  | Fdecl(f)-> txt_of_fdecl f
  | Cake(fdecl, args) -> sprintf "Cake(%s, [%s])"
      (txt_of_expr fdecl) (txt_of_list args)
  | If(e1, e2, e3) -> sprintf "If(%s, %s, %s)"
      (txt_of_expr e1) (txt_of_expr e2) (txt_of_expr e3)

and txt_of_dist d =
  sprintf "Dist({ min=%s ; max=%s ; dist_func=%s })"
    (txt_of_expr d.min) (txt_of_expr d.max) (txt_of_expr d.dist_func)

and txt_of_discr_dist d =
  sprintf "Dist({ vals=%s ; weights=%s })"
    (txt_of_expr d.vals) (txt_of_expr d.weights)

(* Function declarations *)
and txt_of_fdecl f =
  sprintf "Fdecl({ params=[%s] ; body=%s ; return = %s })"
    (String.concat " ; " f.params) (txt_of_stmts f.body) (txt_of_expr f.return)

(* Lists *)
and txt_of_list = function
  | [] -> ""
  | [x] -> txt_of_expr x
  | _ as l -> String.concat " ; " (List.map txt_of_expr l)

(* Statements *)
and txt_of_stmt = function
  | Do(expr) -> sprintf "Do(%s)" (txt_of_expr expr)

and txt_of_stmts stmts =
  let rec aux acc = function
      | [] -> sprintf "[%s]" (String.concat " ; " (List.rev acc))
      | stmt :: tl -> aux (txt_of_stmt stmt :: acc) tl
  in aux [] stmts

(* Program entry point *)
let _ =
  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
  let result = txt_of_stmts program in
  print_endline result
```

_arithmetic.in

```
do 1 + 2.1
do 1 - 2.1
do 1 * 2.1
do 1 / 2.1
do 1 % 2.1
do 1 ** 2.1
do -42
do 1 + -43
do 1 * 2 + 3 ** 4
do 1 / 2 % 3 % 4
do 1 + 2 - 3 / 4
do 1 * (2 + 3)
```

_arithmetic.out

```
[
	Do(Binop(Num_lit(1), Add, Num_lit(2.1))) ;
	Do(Binop(Num_lit(1), Sub, Num_lit(2.1))) ;
	Do(Binop(Num_lit(1), Mult, Num_lit(2.1))) ;
	Do(Binop(Num_lit(1), Div, Num_lit(2.1))) ;
	Do(Binop(Num_lit(1), Mod, Num_lit(2.1))) ;
	Do(Binop(Num_lit(1), Pow, Num_lit(2.1))) ;
	Do(Unop(Sub, Num_lit(42))) ;
	Do(Binop(Num_lit(1), Add, Unop(Sub, Num_lit(43)))) ;
	Do(
		Binop(
			Binop(Num_lit(1), Mult, Num_lit(2)),
			Add,
			Binop(Num_lit(3), Pow, Num_lit(4))
		)
	) ;
	Do(
		Binop(
			Binop(
				Binop(Num_lit(1), Div, Num_lit(2)),
				Mod,
				Num_lit(3)
			),
			Mod,
			Num_lit(4)
		)
	) ;
	Do(
		Binop(
			Binop(Num_lit(1), Add, Num_lit(2)),
			Sub,
			Binop(Num_lit(3), Div, Num_lit(4))
		)
	) ;
	Do(
		Binop(
```

```
                    Num_lit(1),
                    Mult,
                    Binop(Num_lit(2), Add, Num_lit(3))
                )
        )
]
```

_cake.in

```
do (() -> return 42)()
do print(
    ((x) -> return x + 1)(41)
)
```

_cake.out

```
[
    Do(
        Cake(
            Fdecl({ params=[] ; body=[] ; return = Num_lit(42) }),
            []
        )
    ) ;
    Do(
        Call(Id(print),
        [
            Cake(
                Fdecl({
                    params=[x] ;
                    body=[] ;
                    return = Binop(Id(x), Add, Num_lit(1))
                }),
                [Num_lit(41)]
            )
        ])
    )
]
```

_conditional.in

```
do if true then 42 else 43
do if false then 42 else 43
do if 42 then "pass" else "fail"
do if 0 then "pass" else "fail"
do if 5 - 5 then true else false
do if true then 40 + 2 else 5 * 5
```

_conditional.out

```
[
    Do(If(Bool_lit(true), Num_lit(42), Num_lit(43))) ;
```

```
    Do(If(Bool_lit(false), Num_lit(42), Num_lit(43))) ;
    Do(If(Num_lit(42), String_lit(pass), String_lit(fail))) ;
    Do(If(Num_lit(0), String_lit(pass), String_lit(fail))) ;
    Do(
        If(
            Binop(Num_lit(5), Sub, Num_lit(5)),
            Bool_lit(true),
            Bool_lit(false)
        )
    ) ;
    Do(
        If(
            Bool_lit(true),
            Binop(Num_lit(40), Add, Num_lit(2)),
            Binop(Num_lit(5), Mult, Num_lit(5))
        )
    )
]
```

_dist.in

```
do <2, 3> | hello |
do x = <5 + 6, 7 + 8> | (x) -> return x * x |

do x <+> x
do x <*> x
do x |+ 2
do x |* 2
do x |** 2
do x <> 100
```

_dist.out

```
[
    Do(
        Dist({
            min=Num_lit(2);
            max=Num_lit(3);
            dist_func=Id(hello)
        })
    );
    Do(
        Assign(
            x,
            Dist({
                min=Binop(Num_lit(5), Add, Num_lit(6));
                max=Binop(Num_lit(7), Add, Num_lit(8));
                dist_func=
                    Fdecl({
                        params=[x] ;
                        body=[] ;
                        return=Binop(Id(x), Mult, Id(x))
```

```
                    })
            })
        )
    );
    Do(Binop(Id(x), D_Plus, Id(x)));
    Do(Binop(Id(x), D_Times, Id(x)));
    Do(Binop(Id(x), D_Shift, Num_lit(2)));
    Do(Binop(Id(x), D_Stretch, Num_lit(2)));
    Do(Binop(Id(x), D_Power, Num_lit(2)));
    Do(Binop(Id(x), D_Sample, Num_lit(100)))
]
```

_func_decl.in

```
do sum = (x, y) ->
    do print(x)
    do print(y)
    return x + y
```

_func_decl.out

```
[
    Do(
        Assign(
            sum,
            Fdecl({
                params=[x ; y] ;
                body=[
                    Do(Call(Id(print), [Id(x)])) ;
                    Do(Call(Id(print), [Id(y)]))
                ] ;
                return=Binop(Id(x), Add, Id(y))
            })
        )
    )
]
```

_id_call.in

```
do PI
do myvar
do print(40 + 2)
do print("fourty-two")
do myfunc(arg1, arg2)
do noargs()
do prec(myvar, 2 * (2 + 3))
```

_id_call.out

```
[
    Do(Id(PI)) ;
```

```
    Do(Id(myvar)) ;
    Do(Call(Id(print), [Binop(Num_lit(40), Add, Num_lit(2))])) ;
    Do(Call(Id(print), [String_lit(fourty-two)])) ;
    Do(Call(Id(myfunc), [Id(arg1) ; Id(arg2)])) ;
    Do(Call(Id(noargs), [])) ;
    Do(
        Call(
            Id(prec),
            [
                Id(myvar) ;
                Binop(Num_lit(2), Mult, Binop(Num_lit(2), Add, Num_lit(3)))
            ]
        )
    )
]
```

_list.in

```
do []
do [1]
do [1, 2, 3]
do [42.0, 42.5]
do ["hello", "world"]
do [true, false, true]
```

_list.out

```
[
    Do(LDecl([])) ;
    Do(LDecl([Num_lit(1)])) ;
    Do(LDecl([Num_lit(1) ; Num_lit(2) ; Num_lit(3)])) ;
    Do(LDecl([Num_lit(42.) ; Num_lit(42.5)])) ;
    Do(LDecl([String_lit(hello) ; String_lit(world)])) ;
    Do(LDecl([Bool_lit(true) ; Bool_lit(false) ; Bool_lit(true)]))
]
```

_literal.in

```
do 42
do 42.1
do "forty-two"
do true
do false
do void
```

_literal.out

```
[
    Do(Num_lit(42)) ;
    Do(Num_lit(42.1)) ;
    Do(String_lit(forty-two)) ;
```

```
    Do(Bool_lit(true)) ;
    Do(Bool_lit(false)) ;
    Do(Void_lit)
]
```

_logical.in

```
do !true
do !false
do true && false
do true || false
do !(true || false)
```

_logical.out

```
[
    Do(Unop(Not, Bool_lit(true))) ;
    Do(Unop(Not, Bool_lit(false))) ;
    Do(Binop(Bool_lit(true), And, Bool_lit(false))) ;
    Do(Binop(Bool_lit(true), Or, Bool_lit(false))) ;
    Do(Unop(Not, Binop(Bool_lit(true), Or, Bool_lit(false))))
]
```

_relational.in

```
do 1 == 1
do 41 != 42
do 41 < 42
do 42 <= 42
do 43 > 42
do 43 >= 42
do !(41 < 42)
do !(42 <= 42)
```

_relational.out

```
[
    Do(Binop(Num_lit(1), Eq, Num_lit(1))) ;
    Do(Binop(Num_lit(41), Neq, Num_lit(42))) ;
    Do(Binop(Num_lit(41), Less, Num_lit(42))) ;
    Do(Binop(Num_lit(42), Leq, Num_lit(42))) ;
    Do(Binop(Num_lit(43), Greater, Num_lit(42))) ;
    Do(Binop(Num_lit(43), Geq, Num_lit(42))) ;
    Do(Unop(Not, Binop(Num_lit(41), Less, Num_lit(42)))) ;
    Do(Unop(Not, Binop(Num_lit(42), Leq, Num_lit(42))))
]
```

12.2.3 Compiler Tests

fail_test.sh

```
#!/bin/bash

NC='\033[0m'
CYAN='\033[0;36m'
GREEN='\033[0;32m'
RED='\033[0;31m'

INPUT_FILES="compiler/fail/*.ods"
TMP_FILE=$(mktemp "compiled.XXXXX")
printf "${CYAN}Running compiler 'fail' tests...\n${NC}"

for input_file in $INPUT_FILES; do
    output_file=${input_file/.ods/.out}

    ../odds.sh -r $input_file $TMP_FILE 2>&1 | cmp -s $output_file -
    if [ "$?" -eq 0 ]; then
        printf "%-65s ${GREEN}SUCCESS\n${NC}" "  - checking $input_file..."
    else
        printf "%-65s ${RED}ERROR\n${NC}" "  - checking $input_file..." 1>&2
        rm -f $TMP_FILE
        exit 1
    fi

done

rm -f $TMP_FILE
exit 0
```

_binop_logical.ods

```
do true && 42
```

_binop_logical.out

```
[31mSemantic error[0m:
  Invalid use of binary operator '&&' with types Bool and Num
```

_binop_numeric.ods

```
do 2 * 3 <= true
```

_binop_numeric.out

```
[31mSemantic error[0m:
  Invalid use of binary operator '<=' with types Num and Bool
```

_call_func_param_num.ods

```
do call = (f, x) ->
  return !f(x)

do being_called = () -> return true

do call(being_called, true)
```

_call_func_param_num.out

```
[31mSemantic error[0m:
  Function 'call' expected argument of type Func(Any => Bool) but was passed Func(
=> Bool) instead
```

_call_length.ods

```
do sum = (x, y) -> return x + y
do sum(1, 2, 3)
```

_call_length.out

```
[31mSemantic error[0m:
  Function 'sum' expects 2 argument(s) but was called with 3 instead
```

_call_nonfunc.ods

```
do f = 5
do f("Hello, world!")
```

_call_nonfunc.out

```
[31mSemantic error[0m:
  Attempting to call Num type 'f' as a function
```

_call_types.ods

```
do sum = (x, y) -> return x + y
do sum(1, "hello")
```

_call_types.out

```
[31mSemantic error[0m:
  Function 'sum' expected argument of type Num but was passed String instead
```

_discr_dist.ods

```
do x = |<4, [1,1,1]>|
```

_discr_dist.out

```
[31mSemantic error[0m:
  Invalid distribution with vals type 'Num' and weights type 'List[Num]'
```

_fdecl_anon.ods

```
do get_add = () -> return (x, y) -> return x + y
do sum = get_add()
do sum(40, true)
```

_fdecl_anon.out

```
[31mSemantic error[0m:
  Function 'sum' expected argument of type Num but was passed Bool instead
```

_fdecl_nested.ods

```
do sum = (x, y, z) ->
    do aux = (a, b) -> return a + b
    return aux(x, y) + z

do sum(1, true, 3)
```

_fdecl_nested.out

```
[31mSemantic error[0m:
  Function 'sum' expected argument of type Num but was passed Bool instead
```

_if_cond.ods

```
do if 42 then true else false
```

_if_cond.out

```
[31mSemantic error[0m:
  Expected type Bool but got type Num instead
```

_if_mismatch.ods

```
do if true then 42 else "hello"
```

_if_mismatch.out

```
[31mSemantic error[0m:
  Invalid attempt to use conditional with mismatched types Num and String
```

_illegal_char.ods

```
do x = 42
```

```
do illegal = ~
```

_illegal_char.out

```
[31mSyntax error[0m, line 2 at column 14: illegal character '~'
```

_list_head_types.ods

```
do head([1,2,3]) || true
```

_list_head_types.out

```
[31mSemantic error[0m:
  Invalid use of binary operator '||' with types Num and Bool
```

_list_heterogeneous.ods

```
do ["one", "of", "these", "things", true, "is", "not", "like", "the", "others"]
```

_list_heterogeneous.out

```
[31mSemantic error[0m:
  Invalid element of type Bool in list of type List[String]
```

_list_tail_types.ods

```
do t = tail([1, 2, 3])
do head(t) || true
```

_list_tail_types.out

```
[31mSemantic error[0m:
  Invalid use of binary operator '||' with types Num and Bool
```

_print_return_type.ods

```
do print("Hello, world!") + 42
```

_print_return_type.out

```
[31mSemantic error[0m:
  Invalid use of binary operator '+' with types String and Num
```

_rec_param_change.ods

```
do f = (x) ->
    do f(2)
    return !f(true)
```

_rec_param_change.out

```
[31mSemantic error[0m:
  Function 'f' expected argument of type Num but was passed Bool instead
```

_rec_return_change.ods

```
do f = (x) ->
  do !f(2)
  return 2
```

_rec_return_change.out

```
[31mSemantic error[0m:
  Invalid return type in function 'f':
    type 'Bool' expected to be returned, but type 'Num' returned instead.
```

_unop_not.ods

```
do !(5)
```

_unop_not.out

```
[31mSemantic error[0m:
  Invalid use of unary operator '!' with type Num
```

_unop_sub.ods

```
do -(true)
```

_unop_sub.out

```
[31mSemantic error[0m:
  Invalid use of unary operator '-' with type Bool
```

_var_scope.ods

```
do x = y
```

_var_scope.out

```
[31mSemantic error[0m:
  Variable 'y' is undefined in current scope
```

pass_test.sh

```
#!/bin/bash

NC='\033[0m'
CYAN='\033[0;36m'
```

```
GREEN='\033[0;32m'
RED='\033[0;31m'

INPUT_FILES="compiler/pass/*.ods"
TMP_FILE=$(mktemp "compiled.XXXXX")
printf "${CYAN}Running compiler 'pass' tests...\n${NC}"

for input_file in $INPUT_FILES; do
    python_file=${input_file/.ods/.py}
    output_file=${input_file/.ods/.out}

    # compile odds program to temp python file
    ../odds.sh -r $input_file $TMP_FILE

    # if python test file exists, compare them
    if [ -e "$python_file" ]; then
        cmp -s $python_file $TMP_FILE
        if [ "$?" -ne 0 ]; then
            printf "%-65s ${RED}ERROR\n${NC}" "  - checking $python_file..." 1>&2
            rm -f $TMP_FILE
            exit 1
        fi
    fi

    # if test output file exists, compare compiled output to it
    if [ -e "$output_file" ]; then
        python $TMP_FILE | cmp -s $output_file -
        if [ "$?" -ne 0 ]; then
            printf "%-65s ${RED}ERROR\n${NC}" "  - checking $output_file..." 1>&2
            rm -f $TMP_FILE
            exit 1
        fi
    fi

    printf "%-65s ${GREEN}SUCCESS\n${NC}" "  - checking $input_file..."
done

rm -f $TMP_FILE
exit 0
```

_anon_in_scope.ods

```
do anon = 42
do () -> return "Hello, world!"
do print(anon)
```

_anon_in_scope.out

```
42
```

_anon_in_scope.py

```
anon_0 = 42
anon_0
def _anon_1():
    return "Hello, world!"
_anon_1
print(anon_0)
```

_arithmetic.ods

```
do 1 + 2.1
do 1 - 2.1
do 1 * 2.1
do 1 / 2.1
do 1 % 2.1
do 1 ** 2.1
do -42
do 42
do 1 + -43
do 1 * 2 + 3 ** 4
do 1 / 2 % 3 % 4
do 1 + 2 - 3 / 4
do 1 * (2 + 3)

do print(1 + 2.1)
do print(1 - 2.1)
do print(1 * 2.1)
do print(1 / 2.0)
do print(1 % 2.1)
do print(1 ** 2.1)
do print(-42)
do print(42)
do print(1 + -43)
do print(1 * 2 + 3 ** 4)
do print(1 / 2 % 3 % 4)
do print(1 + 2 - 3 / 4)
do print(1 * (2 + 3))
```

_arithmetic.out

```
3.1
-1.1
2.1
0.5
1.0
1.0
-42
42
-42
83
0
3
```

```
5
```

_arithmetic.py

```
(1 + 2.1)
(1 - 2.1)
(1 * 2.1)
(1 / 2.1)
(1 % 2.1)
(1 ** 2.1)
(-42)
42
(1 + (-43))
((1 * 2) + (3 ** 4))
(((1 / 2) % 3) % 4)
((1 + 2) - (3 / 4))
(1 * (2 + 3))
print((1 + 2.1))
print((1 - 2.1))
print((1 * 2.1))
print((1 / 2.))
print((1 % 2.1))
print((1 ** 2.1))
print((-42))
print(42)
print((1 + (-43)))
print(((1 * 2) + (3 ** 4)))
print((((1 / 2) % 3) % 4))
print(((1 + 2) - (3 / 4)))
print((1 * (2 + 3)))
```

_assign_equality.ods

```
do print((x = 5) == (y = 7))
do print(x)
do print(a = b = 42)
do print(a)
do print(b)
```

_assign_equality.out

```
False
5
42
42
42
```

_assign_equality.py

```
x_0 = 5
y_1 = 7
```

```
print((x_0 == y_1))
print(x_0)
b_2 = 42
a_3 = b_2
print(a_3)
print(a_3)
print(b_2)
```

_assignment.ods

```
do x = 4
do x = 5
do print(x)
do y = 6
do print(y)
```

_assignment.out

```
5
6
```

_assignment.py

```
x_0 = 4
x_0
x_1 = 5
x_1
print(x_1)
y_2 = 6
y_2
print(y_2)
```

_cake.ods

```
do (
    () ->
        do print("Hello, world!")
        return 42
)()

do print(
    ((x) -> return x + 1)(41)
)
```

_cake.out

```
Hello, world!
42
```

_cake.py

```
def _anon_0():
    print("Hello, world!")
    return 42
_anon_0()
def _anon_1(x_2):
    return (x_2 + 1)
print(_anon_1(41))
```

_cons_overwrite.ods

```
do x = [1, 2, 3]
do cons = 42
do print(0 :: x)
do print(cons)
```

_cons_overwrite.out

```
[0, 1, 2, 3]
42
```

_cons_overwrite.py

```
x_0 = [1, 2, 3]
x_0
cons_1 = 42
cons_1
print(([0] + x_0))
print(cons_1)
```

_fdecl.ods

```
do call = (x, y) ->
  do print("Hello, world!")
  return x + y

do no_args = () -> return void

do x = call(40, 2)
do print(x)
do no_args()
```

_fdecl.out

```
Hello, world!
42
```

_fdecl.py

```
def call_0(x_1, y_2):
```

```
    print("Hello, world!")
    return (x_1 + y_2)
call_0
def no_args_3():
    return None
no_args_3
x_4 = call_0(40, 2)
x_4
print(x_4)
no_args_3()
```

_fdecl_anon.ods

```
do () -> return "hello"

do y = (x) -> return x() + 1

do print(
    y(() -> return 41)
)
```

_fdecl_anon.out

```
42
```

_fdecl_anon.py

```
def _anon_0():
    return "hello"
_anon_0
def y_1(x_2):
    return (x_2() + 1)
y_1
def _anon_3():
    return 41
print(y_1(_anon_3))
```

_fdecl_equality.ods

```
do foo = () -> return 42
do print((bar = () -> return "hi") == foo)
do print(foo == foo)
do print((baz = () -> return true) == baz)
```

_fdecl_equality.out

```
False
True
True
```

_fdecl_equality.py

```
def foo_0():
    return 42
foo_0
def bar_1():
    return "hi"
print((bar_1 == foo_0))
print((foo_0 == foo_0))
def baz_2():
    return True
print((baz_2 == baz_2))
```

_fdecl_nested.ods

```
do foo = (x, y) ->
  do bar = (a, b) ->
    do print("Hello, world!")
    return a + b
  do z = bar(1, 2)
  return x + z

do y = foo(39, 100)
do print(y)
```

_fdecl_nested.out

```
Hello, world!
42
```

_fdecl_nested.py

```
def foo_0(x_1, y_2):
    def bar_3(a_4, b_5):
        print("Hello, world!")
        return (a_4 + b_5)
    bar_3
    z_6 = bar_3(1, 2)
    z_6
    return (x_1 + z_6)
foo_0
y_7 = foo_0(39, 100)
y_7
print(y_7)
```

_hello_world.ods

```
do print("Hello, world!")
```

_hello_world.out

```
Hello, world!
```

_hello_world.py

```
print("Hello, world!")
```

_id_call.ods

```
/* PI; */
/* EUL; */

do print(40 + 2)
do print("fourty-two")
do print(2 * (2 + 3))
```

_id_call.out

```
42
fourty-two
10
```

_id_call.py

```
print((40 + 2))
print("fourty-two")
print((2 * (2 + 3)))
```

_if.ods

```
do if true then 42 else 41

do x = if true then () -> return "hello" else () -> return "bye"

do print(x())

do print(if true then "Hello, world!" else "hi there!")

do outer = () ->
    do inner = (x) -> return if x then "true" else "false"
    return inner(true)
do print(outer())
```

_if.out

```
hello
Hello, world!
true
```

_if.py

```
def _cond_0():
    if True:
        return 42
    return 41
_cond_0()
def _anon_1():
    return "hello"
def _anon_2():
    return "bye"
def _cond_3():
    if True:
        return _anon_1
    return _anon_2
x_4 = _cond_3()
x_4
print(x_4())
def _cond_5():
    if True:
        return "Hello, world!"
    return "hi there!"
print(_cond_5())
def outer_6():
    def inner_7(x_8):
        def _cond_10():
            if x_8:
                return "true"
            return "false"
        return _cond_10()
    inner_7
    return inner_7(True)
outer_6
print(outer_6())
```

_list.ods

```
do []
do [1]
do [1, 2, 3]
do [42.0, 42.5]
do ["hello", "world"]
do [true, false, true]

do print([])
do print([1])
do print([1, 2, 3])
do print([42.0, 42.5])
do print(["Hello, ", "world!"])
do print([true, false, true])
```

_list.out

```
[]
[1]
[1, 2, 3]
[42.0, 42.5]
['Hello, ', 'world!']
[True, False, True]
```

_list.py

```
[]
[1]
[1, 2, 3]
[42., 42.5]
["hello", "world"]
[True, False, True]
print([])
print([1])
print([1, 2, 3])
print([42., 42.5])
print(["Hello, ", "world!"])
print([True, False, True])
```

_list_ops.ods

```
do ints = [1, 2, 3, 4]

/* list head tests */
do x = head(ints)
do print(x)
do print(x + 1)

/* list tail tests */
do print(tail(["hi"]))
do y = tail(ints)
do print(y)

/* list length test */
do print(len([]))
do print(len(ints))

/* list cons test */
do z = 5 :: ints
do print(z)
```

_list_ops.out

```
1
2
[]
[2, 3, 4]
```

```
0
4
[5, 1, 2, 3, 4]
```

_list_ops.py

```
ints_0 = [1, 2, 3, 4]
ints_0
x_1 = ints_0[0]
x_1
print(x_1)
print((x_1 + 1))
print(["hi"][1:])
y_2 = ints_0[1:]
y_2
print(y_2)
print(len([]))
print(len(ints_0))
z_3 = ([5] + ints_0)
z_3
print(z_3)
```

_literal.ods

```
do 42
do 42.1
do "forty-two"
do void

do print(42)
do print(42.1)
do print("forty-two")
do print(void)
```

_literal.out

```
42
42.1
forty-two
None
```

_literal.py

```
42
42.1
"forty-two"
None
print(42)
print(42.1)
print("forty-two")
print(None)
```

_logical.ods

```
do !true
do !false
do true && false
do true || false
do !(true || false)

do print(!true)
do print(!false)
do print(true && false)
do print(true || false)
do print(!(true || false))
```

_logical.out

```
False
True
False
True
False
```

_logical.py

```
(not True)
(not False)
(True and False)
(True or False)
(not (True or False))
print((not True))
print((not False))
print((True and False))
print((True or False))
print((not (True or False)))
```

_relational.ods

```
do !true
do !false
do 1 == 1
do 41 != 42
do 41 < 42
do 42 <= 42
do 43 > 42
do 43 >= 42
do !(41 < 42)
do !(42 <= 42)

do print(!true)
do print(!false)
```

```
do print(1 == 1)
do print(41 != 42)
do print(41 < 42)
do print(42 <= 42)
do print(43 > 42)
do print(43 >= 42)
do print(!(41 < 42))
do print(!(42 <= 42))
```

_relational.out

```
False
True
True
True
True
True
True
True
False
False
```

_relational.py

```
(not True)
(not False)
(1 == 1)
(41 != 42)
(41 < 42)
(42 <= 42)
(43 > 42)
(43 >= 42)
(not (41 < 42))
(not (42 <= 42))
print((not True))
print((not False))
print((1 == 1))
print((41 != 42))
print((41 < 42))
print((42 <= 42))
print((43 > 42))
print((43 >= 42))
print((not (41 < 42)))
print((not (42 <= 42)))
```

12.2.4 Library Tests

lib_test.sh

```
#!/bin/bash
```

```
NC='\033[0m'
CYAN='\033[0;36m'
GREEN='\033[0;32m'
RED='\033[0;31m'

INPUT_FILES="lib/*.ods"
TMP_FILE=$(mktemp "compiled.XXXXX")
printf "${CYAN}Running compiler 'lib' tests...\n${NC}"

for input_file in $INPUT_FILES; do
    output_file=${input_file/.ods/.out}

    # compile odds program to temp python file
    ../odds.sh -c $input_file $TMP_FILE

    # if test output file exists, compare compiled output to it
    if [ -e "$output_file" ]; then
        python $TMP_FILE 2>&1 | cmp -s $output_file -
        if [ "$?" -ne 0 ]; then
            printf "%-65s ${RED}ERROR\n${NC}" "  - checking $output_file..." 1>&2
            rm -f $TMP_FILE
            exit 1
        fi
    fi

    printf "%-65s ${GREEN}SUCCESS\n${NC}" "  - checking $input_file..."
done

rm -f $TMP_FILE
exit 0
```

_discr_dist.ods

```
do x = |<[1, 2, 3, 4, 5, 6], [1, 1, 1, 1, 1, 1]>|
do f = (x) -> return 1
do y = <1, 5> | f |
do x <*> y
```

_dist.ods

```
do square = (x) -> return x * x
do x = <0, 1> | square |

do y = <5 + 6, 7 + 8> | (x) -> return x * x |
```

_dist_lib.ods

```
do d1 = <0, 1> | uniform |
do d2 = <0, 2> | uniform |
```

```
do d1 <+> d2
do d1 <*> d2
do d1 |+ 2
do d1 |* 2
do d1 |** 2
do d1 <> 2
```

_exception.ods

```
do exception("My custom exception!")
do print("dead codepath")
```

_exception.out

```
My custom exception!
```

_list_concat.ods

```
do print(list_concat([], []))
do print(list_concat(["Hello, "], ["world!"]))
do print(list_concat([1, 2], [3, 4]))
```

_list_concat.out

```
[]
['Hello, ', 'world!']
[1, 2, 3, 4]
```

_list_empty.ods

```
do nums = [1, 2, 3]
do print(list_empty([]))
do print(list_empty(nums))
```

_list_empty.out

```
True
False
```

_list_fold.ods

```
do nums = [1, 2, 3, 4, 5]
do xor = (a, b) -> return (a || b) && !(a && b)

do print(list_fold((a, x) -> return a + x, 0, []))
do sum = list_fold((a, x) -> return a + x, 0, nums)
do print(sum)

do print(list_fold(xor, true, [false, true, false, true]))
do print(list_fold(xor, true, [false, true, true, true]))
```

_list_fold.out

```
0
15
True
False
```

_list_get.ods

```
do nums = [1, 2, 3, 4, 5]
do print(list_get(2, nums))
do print(list_get(4, nums))
```

_list_get.out

```
3
5
```

_list_get_error.ods

```
do list_get(2, [])
```

_list_get_error.out

```
list_get: List index out of range
```

_list_insert.ods

```
do nums = [1, 2, 3, 4]

do print(list_insert(5, 2, nums))
do print(list_insert(0, 0, nums))
do print(list_insert(5, 4, nums))
do print(list_insert("hello", 0, []))
```

_list_insert.out

```
[1, 2, 5, 3, 4]
[0, 1, 2, 3, 4]
[1, 2, 3, 4, 5]
['hello']
```

_list_insert_error.ods

```
do list_insert(42, 1, [])
```

_list_insert_error.out

```
list_insert: List index out of range
```

_list_iter.ods

```
do nums = ["Hello,", "world!"]

do list_iter(print, [])
do list_iter(print, nums)
```

_list_iter.out

```
Hello,
world!
```

_list_make.ods

```
do print(list_make(0, 0))
do print(list_make(3, 42))
do print(list_make(2, "hi"))
```

_list_make.out

```
[]
[42, 42, 42]
['hi', 'hi']
```

_list_map.ods

```
do bools = [true, false, true, false]
do nums = [1, 2, 3, 4, 5]
do square = (x) -> return x ** 2

do print(list_map(square, []))
do print(list_map(square, nums))
do print(list_map(
    (b) -> return !b,
    bools
))
```

_list_map.out

```
[]
[1, 4, 9, 16, 25]
[False, True, False, True]
```

_list_remove.ods

```
do nums = [0, 1, 2, 3]

do print(list_remove(0, nums))
```

```
do print(list_remove(2, nums))
do print(list_remove(3, nums))
```

_list_remove.out

```
[1, 2, 3]
[0, 1, 3]
[0, 1, 2]
```

_list_remove_error.ods

```
do list_remove(4, [0, 1, 2, 3])
```

_list_remove_error.out

```
list_remove: List index out of range
```

_list_rev.ods

```
do empty = []
do nums = [1, 2, 3, 4, 5]

do print(list_rev(empty))
do print(list_rev(nums))
do print(list_rev(["world!", "Hello,"]))
```

_list_rev.out

```
[]
[5, 4, 3, 2, 1]
['Hello,', 'world!']
```

_pi_eul.ods

```
do print(PI)
do print(EUL)
```

_pi_eul.out

```
3.14159265359
2.71828182846
```

_print_return.ods

```
do x = print("Hello, world!")
do y = print(42)
do z = print([1, 2, 3])

do print(x)
do print(y)
```

```
do print(z)
```

_print_return.out

```
Hello, world!
42
[1, 2, 3]
Hello, world!
42
[1, 2, 3]
```

12.3 Git Project Log

The following git project log documents our 757 commits spanning from September 23rd through December 21st. Each member put in a considerable amount of time to the project over the course of the semester, checking in code at all hours of the day and night.

```
031c9ed Mon Dec 21 00:21:29 2015 -0500 Alex Kalicki : Merge pull request #150 from odds-lang/dist_shift
3ff9fea Mon Dec 21 00:18:25 2015 -0500 Alex Kalicki : Merge branch 'master' into dist_shift
85ee0a3 Mon Dec 21 00:18:19 2015 -0500 Alex Kalicki : Merge pull request #149 from
odds-lang/example_for_slides
a04176b Mon Dec 21 00:16:41 2015 -0500 Alex Kalicki : fix dist_shift name and order
562ab48 Sun Dec 20 20:25:25 2015 -0500 dannyech : removed example i wasn't using
2a4b772 Sun Dec 20 20:24:25 2015 -0500 dannyech : updated a few examples for slides
d4e34a4 Sun Dec 20 17:31:59 2015 -0500 Alexandra Medway : Merge pull request #148 from
odds-lang/fix_private_funcs
0cba464 Sun Dec 20 17:28:56 2015 -0500 Alex Kalicki : Merge branch 'master' into fix_private_funcs
37c7ada Sun Dec 20 17:28:49 2015 -0500 Alex Kalicki : Merge pull request #147 from
odds-lang/report_correct_line_on_error
fa4bba0 Sun Dec 20 17:16:24 2015 -0500 Alex Kalicki : Merge branch 'line' into
report_correct_line_on_error
4a7757d Sun Dec 20 17:14:18 2015 -0500 Alex Kalicki : update illegal char test to highlight failing
behavior
0e97e52 Sun Dec 20 17:04:38 2015 -0500 Alex Kalicki : fix private function behavior to prevent
overwriting
68e3dab Sun Dec 20 16:34:51 2015 -0500 dannyech : updated fail test output
92f81bd Sun Dec 20 16:31:35 2015 -0500 dannyech : removed line error report for Semantic Errors ONLY
2369e41 Sun Dec 20 16:22:46 2015 -0500 dannyech : error reports now contain correct line
67f55ea Sun Dec 20 14:29:08 2015 -0500 Alex Kalicki : Merge pull request #146 from
odds-lang/sugar_overwrite
d945c05 Sun Dec 20 13:36:36 2015 -0500 Alex Kalicki : Merge branch 'master' into sugar_overwrite
052c1b2 Sun Dec 20 13:33:37 2015 -0500 Alex Kalicki : Merge pull request #145 from
odds-lang/string_funcs
991d31b Sun Dec 20 13:10:46 2015 -0500 Alex Kalicki : Resolve #132. Sugar to builtins, update list lib,
prevent overwrite
5e9827c Sun Dec 20 12:45:23 2015 -0500 Alex Kalicki : remove str concat function
1a97c35 Sun Dec 20 12:40:52 2015 -0500 Alex Kalicki : Merge pull request #143 from
odds-lang/parser_error_reporting
a133752 Sun Dec 20 12:37:49 2015 -0500 Alex Kalicki : update fail tests to use new line number syntax
da160dd Sun Dec 20 12:09:48 2015 -0500 dannyech : finished error reporting messages
b1ed8e6 Sun Dec 20 11:46:50 2015 -0500 Danny : fixed typo
```

```
b1677e2 Sun Dec 20 00:49:50 2015 -0500 dannyech : now reporting line number of errors. Catching and
reporting parse errors
617830e Sat Dec 19 23:04:30 2015 -0500 Danny : Merge pull request #141 from
odds-lang/analyzer_double_unconstrain_bug
1e6e27c Sat Dec 19 23:01:55 2015 -0500 dannyech : temp
5a5f124 Sat Dec 19 21:59:28 2015 -0500 dannyech : removed unnecessary logic from binop
a8eba34 Sat Dec 19 21:50:49 2015 -0500 Danny : Merge pull request #139 from odds-lang/update-precedence
60f8210 Sat Dec 19 20:46:44 2015 -0500 dannyech : Merge branch 'master' into update-precedence
5bdb8c1 Sat Dec 19 20:46:13 2015 -0500 Danny : Merge pull request #140 from odds-lang/dist_example
fd7b336 Sat Dec 19 20:40:59 2015 -0500 dannyech : Fixed bug where tried to constrain function whose
return type is Any upon its invocation
4d43806 Sat Dec 19 19:56:07 2015 -0500 Alexandra Medway : Made changes to electronic devices example
55dbda2 Sat Dec 19 17:26:54 2015 -0500 Alexandra Medway : Small changes
17f5794 Sat Dec 19 16:29:21 2015 -0500 dannyech : fixed analyzer is_list_of_num, fixed typo in example
8e8cb2b Sat Dec 19 15:03:19 2015 -0500 dannyech : fixed precedence to match python's
8399352 Sat Dec 19 14:52:54 2015 -0500 dannyech : updated precedence
e136cdf Sat Dec 19 14:24:04 2015 -0500 Alexandra Medway : Added lottery problem
b9832d8 Sat Dec 19 13:57:59 2015 -0500 Alexandra Medway : Merge branch 'master' of
https://github.com/odds-lang/odds into dist_example
1490777 Sat Dec 19 13:57:25 2015 -0500 Danny : Merge pull request #138 from
odds-lang/generic_merge_sort
be8d203 Sat Dec 19 13:53:01 2015 -0500 dannyech : updated merge_sort to work with lists
9baedd3 Sat Dec 19 13:30:20 2015 -0500 Danny : Merge pull request #137 from
odds-lang/fixinig_odds_arithmetic
05df9b3 Sat Dec 19 13:27:15 2015 -0500 dannyech : fixed division so that always floating point
b8b7a8a Sat Dec 19 13:13:28 2015 -0500 Alexandra Medway : Added electronic_device example
161de28 Sat Dec 19 12:20:43 2015 -0500 Alexandra Medway : Merge pull request #136 from
odds-lang/inference_and_checking_examples
685b81a Fri Dec 18 23:07:07 2015 -0500 dannyech : minor edits per @akalicki suggestions
829d23a Fri Dec 18 21:18:15 2015 -0500 dannyech : Added merge_sort, optimized lib plotting, made
IF-THEN-ELSE right associative, fixed 3 bugs in Analyzer
6cd1363 Fri Dec 18 18:03:51 2015 -0500 dannyech : preliminary analyzer examples
7bb1980 Fri Dec 18 13:55:30 2015 -0500 Danny : Merge pull request #135 from
odds-lang/add_docs_folder_update_readme
6a19c1f Fri Dec 18 13:51:52 2015 -0500 dannyech : added simple constraining example
ec90373 Fri Dec 18 13:49:12 2015 -0500 dannyech : fixed typo
b1946fa Fri Dec 18 13:48:50 2015 -0500 dannyech : Merge branch 'master' into
add_docs_folder_update_readme
841e6c5 Fri Dec 18 13:48:35 2015 -0500 dannyech : Revert "fixed typo"
df9534c Fri Dec 18 13:48:27 2015 -0500 dannyech : fixed typo
648e6ad Fri Dec 18 13:47:14 2015 -0500 dannyech : updated readme, made docs and examples folder, moved
constraint system info
bc856d9 Fri Dec 18 13:43:10 2015 -0500 dannyech : Revert "Updated Readme, made Docs folder, made
examples folder"
cbde195 Fri Dec 18 13:40:12 2015 -0500 dannyech : Updated Readme, made Docs folder, made examples
folder
da36ea7 Fri Dec 18 02:32:34 2015 -0500 Alexandra Medway : Merge pull request #124 from
odds-lang/better_syntactic_sugar
b5541ef Fri Dec 18 02:29:28 2015 -0500 Alexandra Medway : Nit by alex
abc0038 Fri Dec 18 02:28:08 2015 -0500 Alexandra Medway : Merge branch 'better_syntactic_sugar' of
https://github.com/odds-lang/odds into better_syntactic_sugar
d227686 Fri Dec 18 02:27:57 2015 -0500 Alexandra Medway : Small changes again
a568192 Fri Dec 18 02:25:17 2015 -0500 dannyech : uncommented test
4d462e1 Fri Dec 18 02:19:30 2015 -0500 Alexandra Medway : Comments and changes suggested by Alexandra
and Alex
2c5fb0a Fri Dec 18 02:12:43 2015 -0500 Alexandra Medway : Merge branch 'master' of
https://github.com/odds-lang/odds into better_syntactic_sugar
183ca34 Fri Dec 18 02:12:24 2015 -0500 Alexandra Medway : Added parserize for discr_dist
e4b07af Fri Dec 18 02:07:36 2015 -0500 Alexandra Medway : Merge pull request #133 from
odds-lang/exeception_fix
6f8101a Fri Dec 18 02:03:32 2015 -0500 Alexandra Medway : Added exceptions, fixed tests
b998bcb Fri Dec 18 01:58:59 2015 -0500 Alex Kalicki : Merge pull request #131 from
odds-lang/pip-requirements
46be887 Fri Dec 18 01:55:09 2015 -0500 Alex Kalicki : draw distinction between ocaml and python install
files
```

```
840eeb4 Fri Dec 18 01:52:18 2015 -0500 Alex Kalicki : remove scipy dependency for faster build times
8e6e0f7 Fri Dec 18 01:46:37 2015 -0500 Alex Kalicki : another try at getting pip install working
57c043b Fri Dec 18 01:45:16 2015 -0500 Alexandra Medway : Merge branch 'better_syntactic_sugar' of
https://github.com/odds-lang/odds into better_syntactic_sugar
201b706 Fri Dec 18 01:44:59 2015 -0500 Alexandra Medway : Merge branch 'better_syntactic_sugar' of
https://github.com/odds-lang/odds into better_syntactic_sugar
c66e530 Fri Dec 18 01:44:58 2015 -0500 dannyech : Merge branch 'master' into better_syntactic_sugar
3367dc7 Fri Dec 18 01:44:37 2015 -0500 dannyech : Merge branch 'master' into better_syntactic_sugar
3461d1a Fri Dec 18 01:41:38 2015 -0500 Alex Kalicki : another try at pip installing requirements in
travis
b5488f3 Fri Dec 18 01:35:40 2015 -0500 Alex Kalicki : another pass at pip matplotlib installing
1065069 Fri Dec 18 01:35:08 2015 -0500 Alexandra Medway : Merge pull request #123 from
odds-lang/print_dist
35e93da Fri Dec 18 01:29:06 2015 -0500 Alex Kalicki : add use mirrors line for redundancy
fda6daa Fri Dec 18 01:26:30 2015 -0500 Alex Kalicki : require matplotlib v1.3.1, add python version to
travis yml, add sudo
6084c9b Fri Dec 18 01:25:35 2015 -0500 Danny : Merge pull request #130 from
odds-lang/list_dist_lib_consistency
05dd181 Fri Dec 18 01:24:53 2015 -0500 dannyech : fixed inconsistencies
461d75f Fri Dec 18 01:18:10 2015 -0500 lillyfwang : added requirements file and fixed path
22776e6 Fri Dec 18 01:15:56 2015 -0500 lillyfwang : hi
c1fd6bf Fri Dec 18 01:13:21 2015 -0500 Lilly Wang : Added comment for print_dist
d020682 Fri Dec 18 01:12:38 2015 -0500 Alexandra Medway : Merge pull request #126 from
odds-lang/uniform
9bd9a71 Fri Dec 18 01:09:58 2015 -0500 lillyfwang : second try, using install pip in travis
62f6828 Fri Dec 18 01:07:19 2015 -0500 Alexandra Medway : Removed uniform
3bb787e Fri Dec 18 01:04:43 2015 -0500 lillyfwang : Merge branch 'print_dist' of
https://github.com/odds-lang/odds into pip-requirements
8f15d5f Fri Dec 18 01:04:14 2015 -0500 lillyfwang : moved plt show to utils
b38a744 Fri Dec 18 00:55:44 2015 -0500 Alexandra Medway : Merge pull request #121 from
odds-lang/discrete
ba8e7ea Fri Dec 18 00:55:41 2015 -0500 lillyfwang : added requirements file
aa77e91 Fri Dec 18 00:53:43 2015 -0500 Alexandra Medway : Nits by Alex
f187368 Fri Dec 18 00:52:08 2015 -0500 Alex Kalicki : Merge pull request #125 from odds-lang/anon_fix
34ffee1 Fri Dec 18 00:45:35 2015 -0500 Alex Kalicki : Prevent anonymous functions from overwriting user
variables. Fixes #72.
a3c074b Fri Dec 18 00:43:03 2015 -0500 lillyfwang : trying to see if this fixes travis
440c605 Fri Dec 18 00:36:46 2015 -0500 Danny : Merge pull request #116 from odds-lang/lib_consistency
6ae3967 Fri Dec 18 00:34:04 2015 -0500 dannyech : Merge branch 'better_syntactic_sugar' into
lib_consistency
c0cf5d9 Fri Dec 18 00:29:11 2015 -0500 lillyfwang : only show plot when someone prints
f59c763 Fri Dec 18 00:17:36 2015 -0500 dannyech : Merge branch 'master' into better_syntactic_sugar
63c69d3 Fri Dec 18 00:11:15 2015 -0500 dannyech : fixed dist sample
bb758ef Fri Dec 18 00:08:54 2015 -0500 lillyfwang : Merge branch 'master' of
https://github.com/odds-lang/odds into print_dist
729bee4 Fri Dec 18 00:05:01 2015 -0500 lillyfwang : removed y axis on printing dists
e940ec5 Fri Dec 18 00:03:21 2015 -0500 lillyfwang : changed y axis to percentages
6d65c80 Thu Dec 17 23:57:10 2015 -0500 Alexandra Medway : Fixed ast
e150547 Thu Dec 17 23:45:21 2015 -0500 lillyfwang : display histograms in different windows
635d70f Thu Dec 17 23:38:22 2015 -0500 Alexandra Medway : Make discrete dist working
51bc36f Thu Dec 17 23:36:01 2015 -0500 lillyfwang : added printing multiple histograms at once
c5fe68e Thu Dec 17 23:33:22 2015 -0500 Alexandra Medway : Added fail test
ca9c4e5 Thu Dec 17 23:32:49 2015 -0500 dannyech : update symbols for dist ops
29c0182 Thu Dec 17 23:16:32 2015 -0500 Alex Kalicki : reorganize parser for clarity
13c46bb Thu Dec 17 23:07:04 2015 -0500 Alexandra Medway : merging
8c866c0 Thu Dec 17 23:05:00 2015 -0500 Alexandra Medway : Discrete samples should be working
a9fe82d Thu Dec 17 22:43:59 2015 -0500 lillyfwang : added special printing functionality for dists
b3f97a0 Thu Dec 17 22:38:54 2015 -0500 Alex Kalicki : Merge pull request #118 from odds-lang/pull_up
72a42bb Thu Dec 17 22:23:28 2015 -0500 Alexandra Medway : Initial steps in discrete dist
64022e8 Thu Dec 17 22:12:14 2015 -0500 Alex Kalicki : merge branch master into pull_up
ce9eb9b Thu Dec 17 21:24:54 2015 -0500 dannyech : Merge branch 'master' into better_syntactic_sugar
3d3ca0b Thu Dec 17 21:24:16 2015 -0500 Danny : Merge pull request #120 from
odds-lang/improve_printer_output
5df619e Thu Dec 17 21:08:20 2015 -0500 Alexandra Medway : Fixed sample operator
7ab0cc0 Thu Dec 17 17:04:04 2015 -0500 Alexandra Medway : Merge branch 'lib_consistency' of
```

```
https://github.com/odds-lang/odds into lib_consistency
f21392e Thu Dec 17 16:57:02 2015 -0500 Alexandra Medway : Merge branch 'master' of
https://github.com/odds-lang/odds into better_syntactic_sugar
3c65f05 Thu Dec 17 16:55:24 2015 -0500 Alexandra Medway : Added better E(X) calculation
665afd1 Thu Dec 17 14:13:45 2015 -0500 Alexandra Medway : Added probability operator
7ca4be2 Thu Dec 17 14:08:26 2015 -0500 dannyech : printer now printing lists
e9c15ee Thu Dec 17 13:48:43 2015 -0500 dannyech : Merge branch 'master' into improve_printer_output
404ddd8 Thu Dec 17 13:44:44 2015 -0500 Danny : Merge pull request #91 from odds-lang/dist
7ec5d85 Thu Dec 17 13:34:17 2015 -0500 Alexandra Medway : Tests passing
07eae41 Thu Dec 17 13:33:08 2015 -0500 Alexandra Medway : Removed sugar portion of concat
a313f14 Thu Dec 17 13:18:42 2015 -0500 Alexandra Medway : Concat op
86bda2e Thu Dec 17 13:14:07 2015 -0500 dannyech : Gave ASN higher precedence than return
24e3d7d Thu Dec 17 13:08:42 2015 -0500 dannyech : Merge branch 'master' into dist
6fcf1f1 Thu Dec 17 13:08:05 2015 -0500 dannyech : Merge branch 'master' into improve_printer_output
b6359b7 Thu Dec 17 13:07:27 2015 -0500 dannyech : Merge branch 'master' into pull_up
d2cfc9a Thu Dec 17 13:06:43 2015 -0500 dannyech : Merge remote-tracking branch
'origin/better_syntactic_sugar' into better_syntactic_sugar
0113441 Thu Dec 17 13:06:26 2015 -0500 dannyech : Merge branch 'master' into better_syntactic_sugar
dc264f4 Thu Dec 17 13:02:24 2015 -0500 Danny : Merge pull request #117 from odds-lang/list_empty
3d2a83e Thu Dec 17 01:14:40 2015 -0500 Alex Kalicki : remove unneccessary Past.Empty, add missing
dependencies to Makefile
1324c42 Thu Dec 17 00:58:55 2015 -0500 Alex Kalicki : remove fdecl anon flag. Resolves #113
1d5d38a Thu Dec 17 00:56:45 2015 -0500 Alex Kalicki : pull up all assigns and fdecls, not just
anonymous ones
7e6e080 Thu Dec 17 00:24:11 2015 -0500 Alex Kalicki : Add list_empty function to list stdlib to clean
up code
98d8040 Thu Dec 17 00:14:44 2015 -0500 Alex Kalicki : change Ast.List to Ast.LDecl for consistency.
resolves #115
b6d4401 Thu Dec 17 00:09:05 2015 -0500 Alex Kalicki : fix dist lib argument inconsistency, add dist
sample test. resolves #114.
a529edf Wed Dec 16 20:54:35 2015 -0500 Alexandra Medway : Fixing the sample method
beca103 Wed Dec 16 20:45:20 2015 -0500 Alexandra Medway : sorted
b8a93df Wed Dec 16 20:44:33 2015 -0500 Alexandra Medway : Refactoring Sample
3e24131 Wed Dec 16 20:22:05 2015 -0500 Alexandra Medway : sample in builtin
2d1a716 Wed Dec 16 20:18:00 2015 -0500 Alexandra Medway : Adding sample operator
51f8f7e Wed Dec 16 19:43:05 2015 -0500 Alexandra Medway : Merge branch 'master' of
https://github.com/odds-lang/odds into dist
0e223de Wed Dec 16 15:12:28 2015 -0500 dannyech : took out unnecessary dist binop logic, added
appropriate logic to sugared operators
1ba7f0f Wed Dec 16 14:45:17 2015 -0500 dannyech : Merge branch 'cons_operator' into
better_syntactic_sugar
442cc0f Wed Dec 16 14:43:58 2015 -0500 dannyech : finished cons
3901aed Wed Dec 16 14:29:31 2015 -0500 dannyech : Merge branch 'dist' into cons_operator
d36b27e Wed Dec 16 14:27:59 2015 -0500 dannyech : preliminary changes
ff8198c Wed Dec 16 00:53:09 2015 -0500 Alex Kalicki : remove unnecessary logic from generator
41d2e2a Wed Dec 16 00:43:44 2015 -0500 dannyech : Made edits per @akalicki suggestions
f0dddcf Tue Dec 15 23:58:52 2015 -0500 dannyech : sat_printer now outputs types to terminal in color -
also better printing for anon and caked functions
32245a1 Tue Dec 15 23:07:09 2015 -0500 dannyech : Merge branch 'master' into improve_printer_output
44110d2 Tue Dec 15 23:06:40 2015 -0500 Danny : Merge pull request #112 from
odds-lang/update_constrain_ew_for_lists_and_funcs
7e8dccc Tue Dec 15 22:57:01 2015 -0500 dannyech : Fixed 4 bugs in Analyzer
587073a Tue Dec 15 20:00:08 2015 -0500 dannyech : Merge branch 'master' into improve_printer_output
269263b Mon Dec 14 23:12:01 2015 -0500 Alex Kalicki : add missing scanner and parser tests. todo:
compiler tests
ed2d284 Mon Dec 14 22:40:07 2015 -0500 Alex Kalicki : merge branch master into branch dist
7c1958d Mon Dec 14 22:32:40 2015 -0500 Alex Kalicki : Merge pull request #110 from odds-lang/dist_lib
fdd3a63 Mon Dec 14 22:22:49 2015 -0500 Alex Kalicki : add PI and EUL constaints. Resolves #82
c165591 Sun Dec 13 23:30:57 2015 -0500 Alex Kalicki : Merge branch 'master' of
https://github.com/odds-lang/odds into dist_lib
c8c936b Sun Dec 13 23:09:20 2015 -0500 Alex Kalicki : Merge pull request #106 from odds-lang/list_lib
e26a28d Sun Dec 13 20:59:51 2015 -0500 dannyech : Added a few comments to check_binop and
collect_constraints
dd07003 Sun Dec 13 20:15:43 2015 -0500 Alex Kalicki : remove empty test
7bf39ae Sun Dec 13 20:06:17 2015 -0500 Alex Kalicki : fix overloaded print comment
```

```
c195e99 Sun Dec 13 19:15:55 2015 -0500 dannyech : Removed another unecessary TODO comment
dbe7ae0 Sun Dec 13 19:14:02 2015 -0500 dannyech : Removed TODO comment
b610238 Sun Dec 13 17:24:06 2015 -0500 Alexandra Medway : Nits and changes
148d074 Sun Dec 13 17:10:43 2015 -0500 Alexandra Medway : Pulling
889e766 Sun Dec 13 17:08:58 2015 -0500 Alexandra Medway : s/r conflicts
96c84cc Sun Dec 13 16:25:54 2015 -0500 Alex Kalicki : add dist standard library. depends on #82
c66dc77 Sun Dec 13 16:11:58 2015 -0500 Alex Kalicki : add list_remove to stdlib
b59f6ac Sun Dec 13 16:04:29 2015 -0500 Alex Kalicki : add list_insert to stdlib
14a51f6 Sun Dec 13 15:16:09 2015 -0500 dannyech : Made printer output more readable
398917c Sun Dec 13 13:11:13 2015 -0500 Alex Kalicki : add list fold tests
ced81e0 Sun Dec 13 04:35:06 2015 -0500 Alex Kalicki : add list_fold + tests
6546d69 Sat Dec 12 20:57:44 2015 -0500 Alex Kalicki : add list_concat to stdlib
875499e Sat Dec 12 20:51:04 2015 -0500 Alex Kalicki : add list_make to stdlib
2516b29 Sat Dec 12 20:46:15 2015 -0500 Alex Kalicki : Merge branch 'list_lib' of
https://github.com/odds-lang/odds into list_lib
37300b3 Sat Dec 12 20:46:08 2015 -0500 Alex Kalicki : finish list_get
98aa296 Sat Dec 12 20:36:10 2015 -0500 dannyech : Added two comments to note places where break point
may be if there is a bug in the future
427ec2f Sat Dec 12 20:29:30 2015 -0500 dannyech : Fixed constraining issue where tried to constrain Any
to Unconst
f8fa370 Sat Dec 12 20:15:24 2015 -0500 Alex Kalicki : rename test to NOT BREAK MY FORMATTING @dannyech
!
a8c64d4 Sat Dec 12 20:11:19 2015 -0500 Alex Kalicki : Merge branch 'master' of
https://github.com/odds-lang/odds into list_lib
2921d74 Sat Dec 12 20:11:07 2015 -0500 Alex Kalicki : Merge branch 'list_lib' of
https://github.com/odds-lang/odds into list_lib
3f4ad6a Sat Dec 12 20:10:56 2015 -0500 Alex Kalicki : remove assign void, add print void return, add
exception, add list iter and nth
d983e7e Sat Dec 12 19:41:36 2015 -0500 dannyech : Merge branch 'master' into dist
cf59f5a Sat Dec 12 19:36:24 2015 -0500 dannyech : Merge branch 'master' into improve_printer_output
097e50a Sat Dec 12 19:35:20 2015 -0500 Danny : Merge pull request #93 from
odds-lang/list_builtins_and_constraining
c8c4dee Sat Dec 12 19:33:06 2015 -0500 dannyech : Added dead code path error
c8174e4 Sat Dec 12 19:19:52 2015 -0500 dannyech : Merge branch 'master' into
list_builtins_and_constraining
fd7b3d4 Sat Dec 12 19:18:02 2015 -0500 dannyech : Commented changes to analyzer for returning innocents
a270bf3 Sat Dec 12 18:53:53 2015 -0500 dannyech : Any as return type
0dbec7d Sat Dec 12 18:51:16 2015 -0500 Alexandra Medway : Added sample step
f9f43d9 Sat Dec 12 18:32:54 2015 -0500 Alexandra Medway : Removed special tokens
d58c531 Sat Dec 12 18:23:01 2015 -0500 Alexandra Medway : Added stretch, shift, exp
9688537 Sat Dec 12 17:55:44 2015 -0500 Alex Kalicki : added list map and tests, removed parser
redundancy
2e2e87c Sat Dec 12 17:48:32 2015 -0500 Alexandra Medway : Merge pull request #104 from
odds-lang/dist_constraining
5224aa9 Sat Dec 12 17:22:11 2015 -0500 Alexandra Medway : Added plus
dcd3d09 Sat Dec 12 17:20:02 2015 -0500 Alex Kalicki : allow functions to return unconst. add working
list_rev
a411210 Sat Dec 12 16:15:11 2015 -0500 Alexandra Medway : Merge branch 'dist' of
https://github.com/alexandramedway/odds into dist_constraining
f490a65 Sat Dec 12 16:06:53 2015 -0500 Alexandra Medway : Adding dist plus and dist times
54bef78 Sat Dec 12 15:08:50 2015 -0500 Alex Kalicki : fix list ops generating functions
0226d64 Sat Dec 12 12:04:27 2015 -0500 dannyech : Merge branch 'master' into improve_printer_output
f439988 Fri Dec 11 19:10:05 2015 -0500 Alexandra Medway : Merge branch 'dist' of
https://github.com/alexandramedway/odds into dist
ea83a74 Fri Dec 11 19:09:56 2015 -0500 Alexandra Medway : Adding functions to the lib
87deb2a Fri Dec 11 13:24:53 2015 -0500 Danny : Merge pull request #100 from
odds-lang/fix_constrain_error_on_passed_functions
151678c Fri Dec 11 13:23:20 2015 -0500 dannyech : Merge branch
'fix_constrain_error_on_passed_functions' into dist_constraining
705433a Fri Dec 11 13:23:09 2015 -0500 dannyech : preliminary dist constraining
2165b0e Fri Dec 11 13:01:34 2015 -0500 dannyech : Merge branch 'dist' into dist_constraining
22d67d0 Fri Dec 11 04:57:24 2015 -0500 Alex Kalicki : Merge pull request #101 from odds-lang/if_pull
84aca17 Fri Dec 11 04:49:14 2015 -0500 Alex Kalicki : nit: add newline at end of test file
6b9b869 Thu Dec 10 22:33:16 2015 -0500 Alex Kalicki : Merge branch 'if_pull' of
https://github.com/odds-lang/odds into list_lib
```

```
21e158f Thu Dec 10 22:30:39 2015 -0500 Alex Kalicki : fix pulling up of if in return statements, add
tests
bd1b88d Thu Dec 10 22:08:42 2015 -0500 dannyech : Added fail tests
a4f8e4a Thu Dec 10 21:56:21 2015 -0500 Alex Kalicki : Merge branch
'fix_constrain_error_on_passed_functions' of https://github.com/odds-lang/odds into list_lib
96d1a94 Thu Dec 10 21:48:45 2015 -0500 dannyech : Fixed two erros in constrain system:
47ff24d Thu Dec 10 21:41:04 2015 -0500 Alex Kalicki : Merge branch 'list_builtins_and_constraining'
into list_lib
ac6780a Thu Dec 10 21:40:08 2015 -0500 Alex Kalicki : enable odds stdlib concatenation, add list_rev
test
23a430d Thu Dec 10 21:24:17 2015 -0500 Alex Kalicki : Merge branch 'master' of
https://github.com/odds-lang/odds into list_lib
15dc07a Thu Dec 10 21:20:49 2015 -0500 Alex Kalicki : merge master into dists - fix lib test, remove
broken pipe print (#99) tests
1069422 Thu Dec 10 21:09:10 2015 -0500 dannyech : outline for dist constraining
f994371 Thu Dec 10 20:59:55 2015 -0500 Alex Kalicki : Merge branch 'master' of
https://github.com/odds-lang/odds into list_builtins_and_constraining
e70e9ba Thu Dec 10 20:54:35 2015 -0500 Alex Kalicki : Merge pull request #98 from odds-lang/stdlib
c919e9e Thu Dec 10 16:37:45 2015 -0500 Alex Kalicki : fix compiler tests to use new directory
0bdc31b Thu Dec 10 16:36:25 2015 -0500 Alex Kalicki : add pretty error if not compiled, move executable
to top directory
a875ce3 Thu Dec 10 16:29:24 2015 -0500 Alex Kalicki : fix stdlib location calling, make script more
robust, fix dist code
ab0e3ba Thu Dec 10 14:34:43 2015 -0500 dannyech : Updated printer to print a little more pretty
99d7da9 Thu Dec 10 14:04:16 2015 -0500 Alex Kalicki : begin executable rework
035f357 Thu Dec 10 13:43:03 2015 -0500 Alexandra Medway : Merging with master and updates
d39ea9c Thu Dec 10 13:40:29 2015 -0500 Alexandra Medway : Merge branch 'master' of
https://github.com/alexandramedway/odds into dist
eb16a98 Thu Dec 10 13:40:05 2015 -0500 dannyech : Fixed error in generator that occurred on incorrect
resolving of merge conflict
1b4a6d5 Thu Dec 10 11:20:48 2015 -0500 dannyech : Removed binaries and compiled files
4ec533e Thu Dec 10 01:32:15 2015 -0500 dannyech : Merged pre-messup from list_builtins
097f91e Thu Dec 10 01:04:25 2015 -0500 Alex Kalicki : minor changes
2c3efdc Thu Dec 10 01:03:57 2015 -0500 Alexandra Medway : Merge pull request #90 from
odds-lang/sast_change
d4a95a5 Thu Dec 10 01:01:48 2015 -0500 Alexandra Medway : Typo, again
43dfebe Thu Dec 10 00:53:50 2015 -0500 Alexandra Medway : Typo
28f9c90 Thu Dec 10 00:46:56 2015 -0500 Alex Kalicki : remove print statements
0cbc436 Thu Dec 10 00:41:04 2015 -0500 Alex Kalicki : fixed dist creaton code
59896a4 Thu Dec 10 00:24:13 2015 -0500 Alex Kalicki : fix dist creation function
745cc7b Wed Dec 9 23:36:12 2015 -0500 Alexandra Medway : Tests passing
6b9e982 Wed Dec 9 23:31:25 2015 -0500 Alexandra Medway : No more shift reduce conflicts
5d2b67c Wed Dec 9 23:04:36 2015 -0500 dannyech : fixed readme formatting
b3dd01e Wed Dec 9 23:04:11 2015 -0500 dannyech : Fixed typos in Readme
821bb6d Wed Dec 9 23:03:37 2015 -0500 dannyech : shamed constraint system in readme
238dd49 Wed Dec 9 23:00:31 2015 -0500 Alexandra Medway : Added automatic uniform dists
8f62ffb Wed Dec 9 22:52:53 2015 -0500 dannyech : List constraining now working. Also constraining on
cons().
5386302 Wed Dec 9 22:38:16 2015 -0500 lillyfwang : added test files for dist, need to fix lib_test.sh
cfd06c7 Wed Dec 9 22:22:36 2015 -0500 Alexandra Medway : Tests passing
d600ceb Wed Dec 9 22:22:33 2015 -0500 Alex Kalicki : merge branch master into list_lib, begin list
stdlib
7eee28d Wed Dec 9 22:21:01 2015 -0500 Alexandra Medway : Merging with master
94b6cec Wed Dec 9 22:20:37 2015 -0500 Alexandra Medway : Merge branch 'master' of
https://github.com/alexandramedway/odds into sast_change
86163ec Wed Dec 9 22:17:54 2015 -0500 Alexandra Medway : Moved location of sast_printer and made it
accessible from odds.ml
3093c69 Wed Dec 9 22:02:42 2015 -0500 lillyfwang : Merge branch 'master' of
https://github.com/odds-lang/odds into dist
1a1a0cd Wed Dec 9 22:02:15 2015 -0500 Lilly Wang : Merge pull request #89 from odds-lang/compile_change
ead34d4 Wed Dec 9 21:47:56 2015 -0500 Alexandra Medway : Fixed range
2f1e330 Wed Dec 9 21:43:52 2015 -0500 lillyfwang : added dist parser tests
357dd42 Wed Dec 9 21:40:20 2015 -0500 Alexandra Medway : Moved code generation from generator.ml to
odds.ml
b808469 Wed Dec 9 21:31:20 2015 -0500 Alexandra Medway : Removed file creation functionality from
```

```
generator
d47b3b4 Wed Dec 9 21:24:54 2015 -0500 lillyfwang : more nit changes
903f4f6 Wed Dec 9 21:24:18 2015 -0500 lillyfwang : nit changes, cleaned up code
b8a9bea Wed Dec 9 21:06:08 2015 -0500 lillyfwang : added notes for danny for dist constraining
2ef6773 Wed Dec 9 21:04:53 2015 -0500 Alexandra Medway : Added Std Lib
e9f8299 Wed Dec 9 21:01:17 2015 -0500 lillyfwang : fixed statment passing in dist
5791172 Wed Dec 9 20:32:28 2015 -0500 lillyfwang : fixed merge conflicts
a6f49c5 Wed Dec 9 20:27:55 2015 -0500 lillyfwang : dists work, need to add tests
a05da31 Wed Dec 9 20:23:31 2015 -0500 Alex Kalicki : Merge pull request #88 from odds-lang/caking
09cb52a Wed Dec 9 20:21:17 2015 -0500 Alex Kalicki : merge branch master into caking
9a196d6 Wed Dec 9 18:43:01 2015 -0500 Danny : Merge pull request #85 from odds-lang/constrain_any
988b9bb Wed Dec 9 18:35:22 2015 -0500 dannyech : Merge branch 'master' into constrain_any
ebbed8c Wed Dec 9 18:29:14 2015 -0500 Danny : Merge pull request #83 from odds-lang/conditionals
094c930 Wed Dec 9 17:12:23 2015 -0500 Alex Kalicki : add cake to tokenizer, use standard stmts' syntax
f21a45b Wed Dec 9 17:03:26 2015 -0500 Alex Kalicki : add sast error for cons, will wait on constraining
for input from @dannyech
9c1af30 Wed Dec 9 16:57:02 2015 -0500 Alex Kalicki : add list cons pass tests
5093263 Wed Dec 9 16:49:05 2015 -0500 Alex Kalicki : begin list_op rework to allow constraining on cons
a394dcc Wed Dec 9 16:40:07 2015 -0500 Alex Kalicki : implement list length, begin list cons (need to
check_constraints)
88b3318 Wed Dec 9 15:50:18 2015 -0500 Alex Kalicki : add list head and tail fail tests
a576dff Wed Dec 9 15:46:31 2015 -0500 Alex Kalicki : add list tail functionality
3ccb72f Wed Dec 9 15:30:53 2015 -0500 Alex Kalicki : remove static string pattern matching, use
builtins instead
e3d6951 Wed Dec 9 15:18:30 2015 -0500 Alex Kalicki : implement list head operator
cb5d239 Wed Dec 9 14:25:42 2015 -0500 Alex Kalicki : Merge branch 'constrain_any' into list_builtins
2fcaf12 Wed Dec 9 14:17:24 2015 -0500 Alex Kalicki : merge branch master into list_builtins
9274556 Wed Dec 9 14:15:14 2015 -0500 Alex Kalicki : Merge branch 'master' into constrain_any
ba3277f Wed Dec 9 14:14:51 2015 -0500 Alex Kalicki : Merge branch 'master' into caking
ef0b76a Wed Dec 9 14:14:30 2015 -0500 Alex Kalicki : Merge branch 'master' of
https://github.com/odds-lang/odds into conditionals
473cc27 Wed Dec 9 14:13:51 2015 -0500 Alex Kalicki : remove unmatched, leftover test in master
739a505 Wed Dec 9 14:06:59 2015 -0500 Alex Kalicki : remove extra parser indentation
c2f5881 Wed Dec 9 12:44:02 2015 -0500 dannyech : Add indent for style
1f0fd14 Wed Dec 9 11:28:35 2015 -0500 dannyech : Indented for style
aac4bdd Wed Dec 9 11:23:45 2015 -0500 dannyech : Merge remote-tracking branch 'origin/caking' into
caking
e9531b2 Wed Dec 9 11:22:31 2015 -0500 dannyech : Caked functions should now be parsing correctly
(without any S/R conflicts)
7fb64eb Wed Dec 9 11:00:00 2015 -0500 Alexandra Medway : Removed precedence try
ec563dc Wed Dec 9 10:53:13 2015 -0500 Alexandra Medway : Whitespace
8b0c1ba Wed Dec 9 10:52:15 2015 -0500 Alexandra Medway : Merge branch 'caking' of
https://github.com/alexandramedway/odds into caking
92de86a Wed Dec 9 10:51:59 2015 -0500 Alexandra Medway : Giving function calls precedence
16213c9 Wed Dec 9 03:16:36 2015 -0500 Alex Kalicki : initial pass at caking. resolves #75
72dcfc1 Wed Dec 9 02:15:09 2015 -0500 Alex Kalicki : remove extra test file, add test for vanilla if
statement
99d67c3 Wed Dec 9 02:08:23 2015 -0500 Alex Kalicki : use previously constrained type in check_if
edfb81e Wed Dec 9 02:06:31 2015 -0500 Alex Kalicki : minor fixes
881ba20 Wed Dec 9 02:03:54 2015 -0500 Alex Kalicki : generalize if handling to allow assignment,
pulling up from inside calls
d0b03fd Wed Dec 9 01:40:43 2015 -0500 Alex Kalicki : fix nits
105b15f Wed Dec 9 01:39:26 2015 -0500 Alex Kalicki : minor output changes, extra output tests. add
tests that should pass but make conds fail
67baaa2 Wed Dec 9 00:46:35 2015 -0500 Alex Kalicki : extra fail test, spacing nits, fix compiler
warnings
0b0fc6a Tue Dec 8 10:42:54 2015 -0500 dannyech : Merge branch 'master' into constrain_any
c42be80 Mon Dec 7 22:06:23 2015 -0500 Alexandra Medway : Changes suggested by Alex
2ea818c Mon Dec 7 21:58:01 2015 -0500 Lilly Wang : Merge pull request #86 from
odds-lang/conditionals-refactor
7ec0b4d Mon Dec 7 21:55:02 2015 -0500 lillyfwang : added conditional test
1bca7b1 Mon Dec 7 21:51:57 2015 -0500 Alexandra Medway : Fixed small bug
0e0be7f Mon Dec 7 21:40:23 2015 -0500 lillyfwang : fixed anon function bug
55798cb Mon Dec 7 19:19:37 2015 -0500 lillyfwang : Merge branch 'conditionals' of
https://github.com/odds-lang/odds into conditionals
```

```
4251479 Mon Dec 7 19:19:16 2015 -0500 Alexandra Medway : Sast printer fix
9b383ce Mon Dec 7 19:16:51 2015 -0500 lillyfwang : Merge branch 'conditionals' of
https://github.com/odds-lang/odds into conditionals
f27a821 Mon Dec 7 19:16:38 2015 -0500 lillyfwang : made changes, does not compile
97dc188 Mon Dec 7 19:14:26 2015 -0500 Alexandra Medway : Fixing anonymous functions
96bffa8 Mon Dec 7 18:45:49 2015 -0500 lillyfwang : Merge branch 'master' of
https://github.com/odds-lang/odds into dist
da4495f Mon Dec 7 18:39:09 2015 -0500 lillyfwang : Merge branch 'conditionals' of
https://github.com/odds-lang/odds into conditionals
56e652e Mon Dec 7 18:37:52 2015 -0500 Lilly Wang : Merge pull request #84 from odds-lang/anon_func_test
193c5d3 Mon Dec 7 18:32:56 2015 -0500 lillyfwang : fixed merge conflict
4a2c203 Mon Dec 7 18:29:37 2015 -0500 lillyfwang : Merge branch 'master' of
https://github.com/odds-lang/odds into anon_func_test
4db531b Mon Dec 7 18:29:34 2015 -0500 lillyfwang : changed so it is not unconstrained, returns num
42160fc Mon Dec 7 18:10:13 2015 -0500 Lilly Wang : Added new line
bc25f08 Mon Dec 7 17:52:31 2015 -0500 lillyfwang : added anon function pulling up test
3fa7d42 Mon Dec 7 16:39:54 2015 -0500 dannyech : Merge branch 'master' into constrain_any
44ccd22 Mon Dec 7 14:15:06 2015 -0500 Alex Kalicki : Merge branch 'constrain_any' of
https://github.com/odds-lang/odds into list_builtins
90f5d74 Mon Dec 7 14:03:06 2015 -0500 dannyech : Updated Analyzer so collect_constraints won't fail
when fed list. See description of further info:
2e8138c Mon Dec 7 13:57:51 2015 -0500 Alex Kalicki : Merge pull request #81 from
odds-lang/constrain_return
00b600c Sun Dec 6 19:06:09 2015 -0500 Alexandra Medway : Added to sast printer
a83dc13 Sun Dec 6 18:59:00 2015 -0500 Alexandra Medway : Small modifications
f40be33 Sun Dec 6 18:50:28 2015 -0500 Alexandra Medway : Two types of if statements, if_assign and if
d3657cb Sun Dec 6 18:33:01 2015 -0500 Alexandra Medway : Continuing with conditionals
c5c6f32 Sun Dec 6 17:08:40 2015 -0500 Alexandra Medway : Tests are passing, new past working
8bd27d5 Sun Dec 6 15:42:09 2015 -0500 lillyfwang : added dist to parser
a7979a7 Sun Dec 6 15:24:09 2015 -0500 Alexandra Medway : Debugging
be81b18 Sun Dec 6 15:16:14 2015 -0500 Alexandra Medway : Refactoring the past
5d1f8af Sun Dec 6 15:06:26 2015 -0500 Alex Kalicki : Merge remote-tracking branch
'origin/constrain_any' into list_builtins
14297b3 Sun Dec 6 15:06:05 2015 -0500 Alex Kalicki : begin work on list operators
99bbd32 Sun Dec 6 13:53:17 2015 -0500 Alex Kalicki : Merge branch 'master' of
https://github.com/odds-lang/odds into constrain_return
9710795 Sun Dec 6 13:52:10 2015 -0500 Alex Kalicki : update build image and link in README
20751b2 Sun Dec 6 12:09:47 2015 -0500 dannyech : Added TODO for collect_constraints style
622dc5e Sun Dec 6 12:07:46 2015 -0500 dannyech : Added explanatory comment for new collect_constraints
22d8fee Sat Dec 5 20:47:37 2015 -0500 dannyech : More ground work for conditionals in analyzer
250f407 Sat Dec 5 15:03:15 2015 -0500 Alex Kalicki : update build image and link for migrated repo
d349308 Sat Dec 5 14:54:54 2015 -0500 Alex Kalicki : fix sast bug on return type mismatch. fixes #79
14e96c8 Wed Dec 2 22:10:33 2015 -0500 Alexandra Medway : Finished preliminary conditionals
5493803 Wed Dec 2 21:06:48 2015 -0500 dannyech : Fixed bug where would try to constrain Any to Unconst
and throw error
ac749c3 Wed Dec 2 20:09:50 2015 -0500 dannyech : Fixed issue in collect_constraints where error was
thrown when it should Not be thrown.
450b8a8 Wed Dec 2 19:48:02 2015 -0500 Alexandra Medway : initial steps constraining conditionals
15e6d2c Wed Dec 2 19:08:42 2015 -0500 Alexandra Medway : Merging and updating conditionals
ea003e1 Wed Dec 2 18:48:03 2015 -0500 Alex Kalicki : Merge pull request #77 from alexandramedway/and_or
3576057 Wed Dec 2 18:36:04 2015 -0500 Alex Kalicki : Merge branch 'master' of
https://github.com/alexandramedway/odds into and_or
1277479 Wed Dec 2 18:35:11 2015 -0500 Alex Kalicki : fix merge conflict
75d6ba6 Wed Dec 2 18:32:52 2015 -0500 Alex Kalicki : Merge pull request #74 from alexandramedway/past
47f4a83 Wed Dec 2 18:31:02 2015 -0500 Alex Kalicki : merge master refactor into past
57664f8 Wed Dec 2 18:28:24 2015 -0500 Alex Kalicki : Merge pull request #69 from
alexandramedway/recursive_type_constraining
82883f1 Wed Dec 2 18:25:41 2015 -0500 dannyech : Wrote TODO
54c9b27 Wed Dec 2 12:08:52 2015 -0500 dannyech : Fixed bug where if constraining both function's arg
and return type, only the return type would get constrained.
dd8b779 Wed Dec 2 05:27:39 2015 -0500 Alex Kalicki : remove unneccessary newline at end of defs
3bfc7ae Wed Dec 2 05:25:33 2015 -0500 Lilly Wang : Merge pull request #73 from
alexandramedway/past_function_tests
aa14bfa Wed Dec 2 05:13:20 2015 -0500 lillyfwang : fixed tests to remove excess new lines
639ab2c Wed Dec 2 05:07:03 2015 -0500 lillyfwang : Merge branch 'past' of
```

```
https://github.com/alexandramedway/odds into past_function_tests
0cca6e9 Wed Dec 2 05:06:50 2015 -0500 Alex Kalicki : change indentation to spaces, remove extra
newlines
1cadf7a Wed Dec 2 05:02:27 2015 -0500 lillyfwang : added newlines
29a5c67 Wed Dec 2 04:59:50 2015 -0500 lillyfwang : added functions to compiler test
7ddef90 Wed Dec 2 04:39:18 2015 -0500 Alex Kalicki : Add boolean &&, || functionality. Resolves #58
d399d6d Wed Dec 2 04:26:16 2015 -0500 lillyfwang : added tab to whitespace in scanner
615822c Wed Dec 2 04:07:53 2015 -0500 lillyfwang : added comments....as a tribute to @akalicki
7a9967f Wed Dec 2 04:01:44 2015 -0500 lillyfwang : Merge branch 'past' of
https://github.com/alexandramedway/odds into past
cfbbe7d Wed Dec 2 04:01:28 2015 -0500 lillyfwang : merged generator with past
9c4845a Wed Dec 2 03:52:20 2015 -0500 Alex Kalicki : add comments
ac82cb9 Wed Dec 2 03:48:04 2015 -0500 lillyfwang : Merge branch 'past' of
https://github.com/alexandramedway/odds into past
bcf98ff Wed Dec 2 03:47:29 2015 -0500 lillyfwang : finally compiles
eaf5d4d Wed Dec 2 03:46:03 2015 -0500 Alex Kalicki : bugfix
a744826 Wed Dec 2 03:45:28 2015 -0500 Alex Kalicki : add fail case to pattern matching
f9b7b68 Wed Dec 2 03:36:29 2015 -0500 Alex Kalicki : various minor fixes
8da5201 Wed Dec 2 01:39:14 2015 -0500 dannyech : Now constraining params that are functions and
converting Unconst -> Any when necessary
823ae35 Tue Dec 1 09:57:41 2015 -0500 dannyech : Added some comments, added potential functions to use
for constraining functions that are parameters.
b088bf2 Tue Dec 1 04:40:12 2015 -0500 lillyfwang : fixed more bugs
ef95056 Tue Dec 1 04:18:39 2015 -0500 lillyfwang : Need to fix anon function
e6dbd2d Tue Dec 1 04:06:41 2015 -0500 lillyfwang : Initial pythonizer...not compiling yet
7b1c191 Tue Dec 1 02:19:49 2015 -0500 Alex Kalicki : prevent reassignment of function id in declaration
2ef5109 Tue Dec 1 01:33:45 2015 -0500 Alex Kalicki : fix Not_found lookup error when called function is
a param
abdb093 Mon Nov 30 17:21:10 2015 -0500 Alex Kalicki : merge branch master-refactor into
recursive_type_constraining
ef78bd2 Mon Nov 30 16:31:31 2015 -0500 dannyech : Changed `Array.to_list (Array.make (List.length
a_list) Unconst)` to List.map (fun _ -> Unconst) a_list in two places
ee1d1ae Mon Nov 30 16:18:06 2015 -0500 dannyech : Now constraining functions passed as args
447bf9d Mon Nov 30 16:07:03 2015 -0500 Alex Kalicki : Merge pull request #68 from
alexandramedway/sast-tests
a669667 Mon Nov 30 15:31:35 2015 -0500 Alex Kalicki : add sast fail tests for anonymous and nested
functions
f428451 Mon Nov 30 15:02:55 2015 -0500 dannyech : Added logic to check_fdecl to ensure that:
01a7450 Mon Nov 30 11:45:03 2015 -0500 Alex Kalicki : improved fcall error handling, add fail tests
4d4862e Mon Nov 30 11:19:45 2015 -0500 Alex Kalicki : add a bunch of sast failing tests
a8db3fc Mon Nov 30 11:19:23 2015 -0500 Alex Kalicki : move sast_printer to test/compiler directory, lay
groundwork for future tests
6e8ec2f Mon Nov 30 11:10:51 2015 -0500 Alexandra Medway : No more recursion in indent_of_num
e6c0fbe Mon Nov 30 11:04:20 2015 -0500 Alexandra Medway : Added anon flag in fdecl
3cf8181 Mon Nov 30 10:46:25 2015 -0500 Alexandra Medway : Tests passing
7892388 Mon Nov 30 00:41:12 2015 -0500 dannyech : Fixed a merge error in Scanner
e242264 Sun Nov 29 23:21:33 2015 -0500 dannyech : Merge branch 'master-refactor' into
recursive_type_constraining
e3d82d8 Sun Nov 29 22:32:03 2015 -0500 Alexandra Medway : Final touches, fixed return bug
4c9ba3b Sun Nov 29 22:03:16 2015 -0500 Alexandra Medway : Indentation in generator
b9b0787 Sun Nov 29 21:24:30 2015 -0500 Alexandra Medway : Merge branch 'master-refactor' of
https://github.com/alexandramedway/odds into past
ef6c915 Sun Nov 29 20:44:55 2015 -0500 dannyech : Fixed Sast pretty printer so that it is printing more
~pretty~
b7c3b15 Sun Nov 29 20:29:40 2015 -0500 dannyech : Analyzer now constraining function return types even
on recursive calls
632c5c0 Sun Nov 29 20:09:14 2015 -0500 dannyech : Analyzer now constraining function parameter types on
recursive calls.
68043f5 Sun Nov 29 19:08:41 2015 -0500 dannyech : Added type Any to Sast and made appropriate changes
in Analyzer
2bdda88 Sun Nov 29 18:42:37 2015 -0500 Alex Kalicki : merge branch master into master-refactor
9366ef5 Sun Nov 29 18:40:33 2015 -0500 dannyech : Merge branch 'master-refactor' into
recursive_type_constraining
7abaaf55 Sun Nov 29 18:38:14 2015 -0500 dannyech : Changed check_fdecl so that current func intialy
assigned type Func(Unconst params & Unconst return) instead of just Unconst
```

```
4d006c9 Sun Nov 29 18:37:59 2015 -0500 Alex Kalicki : Merge pull request #62 from
alexandramedway/fail_tests
92a39b5 Sun Nov 29 18:33:46 2015 -0500 Danny : Merge pull request #67 from
alexandramedway/constraint_system
407e757 Sun Nov 29 01:05:56 2015 -0500 Alex Kalicki : fix broken double-constraining case\: do x = 5 do
y = x + 2
b633782 Sun Nov 29 01:01:59 2015 -0500 Alex Kalicki : remove old inference function
69acc9e Sun Nov 29 00:28:50 2015 -0500 Alex Kalicki : allow type constraining in function calls
8111688 Sun Nov 29 00:14:11 2015 -0500 Alex Kalicki : rework fcall arg check logic
264ba15 Sun Nov 29 00:05:16 2015 -0500 Alex Kalicki : fix build error, constrain list types
ddf959c Sat Nov 28 16:43:48 2015 -0500 Alex Kalicki : allow Unconst equality check, add some comments
f559646 Sat Nov 28 16:15:45 2015 -0500 Alex Kalicki : remove superfluous slice_off_ssid function
a195646 Sat Nov 28 14:59:27 2015 -0500 dannyech : Updated Analyzer error reporting messages
615fcfa Sat Nov 28 14:05:23 2015 -0500 dannyech : Added type constraining for Binops
37b52dd Sat Nov 28 12:16:10 2015 -0500 lillyfwang : Very basic beginning of the past
596cb2d Sat Nov 28 01:48:38 2015 -0500 Alex Kalicki : minor naming nits
75cf978 Sat Nov 28 01:46:18 2015 -0500 Alex Kalicki : get unary constraint inference working!
a325586 Sat Nov 28 01:31:50 2015 -0500 Alex Kalicki : lay groundwork for future type inference
5c9c1bd Sat Nov 28 01:04:11 2015 -0500 Alex Kalicki : remove unnecessary tests, char-by-char compare,
fail test tmp file
7e1f2ba Sat Nov 28 00:19:01 2015 -0500 Alexandra Medway : Adding all Changes
9fcccfa Sat Nov 28 00:18:40 2015 -0500 Alexandra Medway : Changes suggested by Alex
f203b31 Fri Nov 27 21:38:19 2015 -0500 Alex Kalicki : add some comments, fix error, evaluate func type
in env. TODO: constraining
8904734 Fri Nov 27 01:52:53 2015 -0500 Alex Kalicki : begin environment reorganization
7758622 Tue Nov 24 13:32:02 2015 -0500 dannyech : Added Exceptions to Analyzer and Scanner. Added Error
handling to sast_printer.
909f3c1 Tue Nov 24 01:33:18 2015 -0500 Alex Kalicki : Merge branch 'constraint_system' of
https://github.com/alexandramedway/odds into constraint_system
e17ae5a Tue Nov 24 01:33:07 2015 -0500 Alex Kalicki : throw error if attempting to assign to void
4ee0eaa Tue Nov 24 00:18:38 2015 -0500 dannyech : Updated sast_printer tabbing mechanism so that it
uses an ocaml String library function (String.make)
563057d Mon Nov 23 22:30:08 2015 -0500 dannyech : sat_printer does tabbing. Types printed now being
with Capital letter (makes it easier to distinguish type from variable name in printer).
b1cd488 Mon Nov 23 19:17:25 2015 -0500 dannyech : nAdded todo: RAISE ERROR IF FUNCTION RETURN TYPE IS
UNCONSTRAINED
d16e12b Mon Nov 23 18:25:55 2015 -0500 Alex Kalicki : add void_lit to sast_printer to remove compiler
warnings
5174527 Mon Nov 23 18:21:39 2015 -0500 Alex Kalicki : add void_lit, set print to return void
0ee179c Mon Nov 23 18:12:50 2015 -0500 Alex Kalicki : nit: don't capture unneeded expression
f5aeb87 Mon Nov 23 17:12:12 2015 -0500 dannyech : Assignment now assigns new variable a type.
28eeb00 Mon Nov 23 15:03:40 2015 -0500 dannyech : sast_printer now working
b873b38 Mon Nov 23 01:18:57 2015 -0500 dannyech : Got rid of sast_printer in top level Makefile
64a05fe Mon Nov 23 00:32:53 2015 -0500 Alex Kalicki : handle empty list case in analyzer. working
build!!!
bd22b56 Mon Nov 23 00:23:11 2015 -0500 Alex Kalicki : fix fcall analyzer error
256bbbe Mon Nov 23 00:15:26 2015 -0500 Alex Kalicki : comment out logical AND and OR - will add to
parser + tests in future PR
1cdb11b Mon Nov 23 00:10:10 2015 -0500 Alex Kalicki : require argument for print, minor makefile
changes
cc0c923 Mon Nov 23 00:03:09 2015 -0500 Alex Kalicki : get sast analyzer compiling!
0f9ad3e Sun Nov 22 17:23:28 2015 -0500 Alexandra Medway : Added failed to test suite
9505dcc Sun Nov 22 17:19:37 2015 -0500 Alexandra Medway : Finished failed test script
02bc2e7 Sun Nov 22 16:27:29 2015 -0500 Alexandra Medway : Adding fail script and a fail test
c4f75fc Sun Nov 22 16:18:34 2015 -0500 Alexandra Medway : Add success directory in test/compiler
26eccd0 Sun Nov 22 16:10:35 2015 -0500 Alexandra Medway : Add expected scanner fail test
186794c Sun Nov 22 16:08:26 2015 -0500 Alexandra Medway : Add scanner error
ea4d2ea Sun Nov 22 15:59:26 2015 -0500 Alexandra Medway : Merge branch 'constraint_system' of
https://github.com/alexandramedway/odds into constraint_system
2524428 Sun Nov 22 15:59:18 2015 -0500 dannyech : Fixed error catching
6a8868d Sun Nov 22 15:59:05 2015 -0500 Alexandra Medway : Merge branch 'constraint_system' of
https://github.com/alexandramedway/odds into constraint_system
20cd18c Sun Nov 22 15:58:16 2015 -0500 dannyech : Added Error handling to sast_printer
d7e593d Sun Nov 22 15:48:37 2015 -0500 Alexandra Medway : Renamed test/sast
bb033c6 Sun Nov 22 15:43:55 2015 -0500 dannyech : Added Make for sast_printer
```

```
12504cf Sun Nov 22 14:48:01 2015 -0500 Alex Kalicki : Revert "Updated sast print to account for the
fact that we are bubbling up type data in the analyzer."
1b4f58d Sun Nov 22 14:40:13 2015 -0500 dannyech : Updated sast print to account for the fact that we
are bubbling up type data in the analyzer.
5f5af22 Sun Nov 22 14:15:52 2015 -0500 dannyech : fixed bin so that the op_str = str_of_binop op. (Was
erroneously  str_of_unop op before)
4377d18 Sun Nov 22 14:13:04 2015 -0500 dannyech : changed decl string print format
61cc449 Sun Nov 22 14:11:32 2015 -0500 dannyech : Finished sast printer - need to debug still
968297f Sun Nov 22 13:42:17 2015 -0500 dannyech : added pattern matchings to str_of_expr (not complete
though)
bdf7eda Sun Nov 22 13:26:29 2015 -0500 dannyech : Preliminary Framework for sast_printer
f417d5d Fri Nov 20 20:10:49 2015 -0500 dannyech : Added List to str_of_type
39c0123 Fri Nov 20 20:00:20 2015 -0500 dannyech : made List type List of data_type in Sast. Made
fdecl_param_error a function in analyzer (was erroneousouly a variable)
c6b3e89 Fri Nov 20 19:45:56 2015 -0500 dannyech : Changed check_decl_return_type to account for fact
that func type is now a record.
f057717 Fri Nov 20 17:18:35 2015 -0500 Alex Kalicki : add fcall TODO comment
ba1a97a Fri Nov 20 17:12:58 2015 -0500 Alex Kalicki : remove all fcall errors
f753dae Fri Nov 20 17:02:14 2015 -0500 Alex Kalicki : add fcall error handling
9114f7f Fri Nov 20 16:42:22 2015 -0500 Alex Kalicki : add most of function calling type checking logic
- TODO: add errors
b8b093d Fri Nov 20 16:22:44 2015 -0500 Alex Kalicki : fdecls only need exprs, combine parser lists,
write func type printer
102e8db Fri Nov 20 15:57:59 2015 -0500 Alex Kalicki : fix sast ordering error
83e1fe7 Fri Nov 20 15:56:21 2015 -0500 Alex Kalicki : rewrite func type as record, add builtins to env
5584ad4 Fri Nov 20 12:19:25 2015 -0500 dannyech : Added Func type to Sast and made check_fdecl return
it. Added decl param error
feee3e1 Fri Nov 20 00:00:41 2015 -0500 dannyech : Added f_decl_param_error for consistency and
extensibility
1d20985 Thu Nov 19 17:29:55 2015 -0500 Alex Kalicki : remove merge conflict text
dc77c54 Thu Nov 19 14:26:41 2015 -0500 dannyech : Merge remote-tracking branch
'origin/constraint_system' into constraint_system
b1fa68e Thu Nov 19 14:25:55 2015 -0500 dannyech : Added function for constraining behavior.
2acecaa Thu Nov 19 13:52:28 2015 -0500 Alex Kalicki : minor formatting + adding todo
4739765 Thu Nov 19 13:41:25 2015 -0500 Alex Kalicki : fix compiler binop pattern matching warning
ed57c1c Thu Nov 19 13:13:25 2015 -0500 dannyech : Renamed StringMap to VarMap because values are not
strings anymore; they are variables
4f1fbdc Thu Nov 19 13:11:22 2015 -0500 dannyech : Combined str_of_unop and str_of_binop into str_of_op
25345a2 Thu Nov 19 13:08:14 2015 -0500 dannyech : Undid one Change @akalicki made to a subroutine in
check_binop
9d3be84 Thu Nov 19 09:59:14 2015 -0500 Alex Kalicki : fix some errors
40e15d4 Wed Nov 18 22:27:24 2015 -0500 dannyech : Finished Sast binop checking. Add logical && and ||
to Ast.
078cd64 Wed Nov 18 21:49:48 2015 -0500 dannyech : Moved error messages to their own section. Continuing
to work on analyzer binop
76b947d Wed Nov 18 21:15:08 2015 -0500 dannyech : Renamed get_type to str_of_type. Changed Unop error
message to show op as well
3ce8a5a Wed Nov 18 20:08:51 2015 -0500 dannyech : Began ground work for type checking. Added type
checking for Unops.
a8a7e79 Wed Nov 18 18:14:09 2015 -0500 Alex Kalicki : Merge pull request #56 from alexandramedway/sast
f7393b7 Wed Nov 18 16:44:37 2015 -0500 Alex Kalicki : variable renaming
c591e11 Wed Nov 18 16:11:43 2015 -0500 Alex Kalicki : force each sast type to maintain its own
environment and explicitly return if changed
74c7f2c Wed Nov 18 16:05:23 2015 -0500 Alex Kalicki : add function id to fdecl env to allow recursive
functions
0129e68 Wed Nov 18 14:59:58 2015 -0500 Alex Kalicki : cleanup
dce7cff Wed Nov 18 13:49:44 2015 -0500 Alex Kalicki : fix function scoping rules, add assignment tests
2eddbb2 Wed Nov 18 04:41:43 2015 -0500 Alex Kalicki : successfully move scoping rules from generator to
sast
7b70d98 Wed Nov 18 01:33:31 2015 -0500 Alex Kalicki : get working (bare-bones) sast inserting between
ast and code generator
fff97d1 Wed Nov 18 00:05:59 2015 -0500 Alex Kalicki : combine int and float_lit
f5dc3ed Tue Nov 17 23:52:10 2015 -0500 Alexandra Medway : deleted extra files
8dd10f5 Tue Nov 17 23:01:06 2015 -0500 Alex Kalicki : separate unops and binops
cc1c1a9 Tue Nov 17 22:54:20 2015 -0500 Alex Kalicki : merge functions branch into sast
```

```
efba120 Tue Nov 17 18:58:09 2015 -0500 dannyech : Preliminary ideas for sast
24f8713 Tue Nov 17 18:17:24 2015 -0500 Alexandra Medway : Begining to change from float to num type
3a7f4d4 Tue Nov 17 00:15:09 2015 -0500 Alex Kalicki : Merge pull request #49 from
alexandramedway/special-ids
38961f9 Mon Nov 16 17:13:44 2015 -0500 Alex Kalicki : raise exception if attempting to look up
undefined key
f72e464 Mon Nov 16 17:08:35 2015 -0500 Alex Kalicki : add PI and EUL to special id list and code
generation
4f290fc Mon Nov 16 12:59:11 2015 -0500 Alex Kalicki : begin to move around id checking
f3af7b9 Sun Nov 15 23:17:49 2015 -0500 Alex Kalicki : use file redirection to bypass cat command
a780dd8 Sun Nov 15 18:40:21 2015 -0500 Alex Kalicki : Merge branch 'master' into functions
f7a6752 Sun Nov 15 18:38:35 2015 -0500 Alex Kalicki : Merge branch 'master' of
https://github.com/alexandramedway/odds
f88bdff Sun Nov 15 18:38:23 2015 -0500 Alex Kalicki : remove pattern match conflict
af79dcf Sun Nov 15 18:35:26 2015 -0500 dannyech : Fixed error in printer.ml
8269fdb Sun Nov 15 18:33:10 2015 -0500 dannyech : Merge branch 'master' into functions
09c2622 Sun Nov 15 18:28:39 2015 -0500 Danny : Merge pull request #48 from alexandramedway/pretty-print
92ca8e3 Sun Nov 15 18:02:16 2015 -0500 Alex Kalicki : add help prompt. Closes #30.
51b39e7 Sun Nov 15 17:45:43 2015 -0500 Alex Kalicki : pass string program instead of stmt list, add raw
printing functionality
4625000 Sun Nov 15 16:49:41 2015 -0500 Alexandra Medway : Fixed fdecl in ast
749248b Sun Nov 15 16:20:13 2015 -0500 Alex Kalicki : nit: fix parser spacing
d78fdca Sun Nov 15 16:18:08 2015 -0500 Alexandra Medway : Initial function creation
93fc36a Sun Nov 15 15:58:52 2015 -0500 Alex Kalicki : change name of process.ml to printer.ml
09f52d0 Sun Nov 15 15:46:23 2015 -0500 Alex Kalicki : fix merge conflicts
c59f1ef Sun Nov 15 15:43:34 2015 -0500 Alex Kalicki : merge master into branch conditionals
5bd28fc Sun Nov 15 15:37:03 2015 -0500 Alex Kalicki : add pretty printer to allow easy indentation
9a8a461 Sun Nov 15 14:57:40 2015 -0500 dannyech : Fixed a few more merge conflicts
f9db990 Sun Nov 15 14:55:41 2015 -0500 dannyech : Merge branch 'master' into functions
7324f26 Sun Nov 15 14:46:52 2015 -0500 Alexandra Medway : Merge pull request #47 from
alexandramedway/assignment_and_scoping
f82e8f4 Sun Nov 15 14:43:37 2015 -0500 Alex Kalicki : cleanup
75bd1c1 Sun Nov 15 14:41:26 2015 -0500 Alex Kalicki : merge master into assignment_and_scoping
e5365b4 Sun Nov 15 14:08:40 2015 -0500 Alex Kalicki : Merge pull request #44 from alexandramedway/lists
7e43868 Sun Nov 15 14:04:12 2015 -0500 Alex Kalicki : resolve list merge conflicts
c4bba8d Sun Nov 15 13:50:08 2015 -0500 Alex Kalicki : Merge pull request #46 from
alexandramedway/parserize-cleanup
bb25d6c Sat Nov 14 13:52:41 2015 -0500 Alex Kalicki : add function parsing and tests, based off scoping
branch
e43013e Sat Nov 14 13:48:47 2015 -0500 Alex Kalicki : remove functions from this branch, will branch
from this one
f86c2cf Fri Nov 13 19:18:52 2015 -0500 Alex Kalicki : clean up parserizer by using sprintf instead of
string concatenation
d6261dd Fri Nov 13 19:01:29 2015 -0500 Alex Kalicki : finish list literal parsing and code generation
57fa937 Fri Nov 13 14:32:13 2015 -0500 Alex Kalicki : merge branch master into assignment_and_scoping
ae6a706 Fri Nov 13 14:27:34 2015 -0500 Alex Kalicki : add generator comments
bce928c Fri Nov 13 14:18:57 2015 -0500 Alex Kalicki : Merge pull request #43 from
alexandramedway/call-id
93cee35 Fri Nov 13 14:05:52 2015 -0500 Alex Kalicki : standardize stmt_list reversing to fix tests
2fabb63 Fri Nov 13 12:21:43 2015 -0500 Alexandra Medway : Test for scoping, changes in parserize to
account for changes in ast
586d8bf Fri Nov 13 11:26:55 2015 -0500 dannyech : Changed Assign to string * expr (In generator,
Parser, and Ast)
93ba1b2 Thu Nov 12 23:15:41 2015 -0500 dannyech : Made Call and Assign pass ID instead of string - as
Alex had originally.
51298d5 Thu Nov 12 22:50:29 2015 -0500 dannyech : Scoping and Assignment now working!
1670d7d Thu Nov 12 22:36:30 2015 -0500 dannyech : Groundwork for scoping. Still not working.
07b2cbc Thu Nov 12 21:01:35 2015 -0500 dannyech : Changed name of of aux function in process_stmt to
do_process_stmts
5d1094f Thu Nov 12 18:57:07 2015 -0500 dannyech : Added framework for list parsing and generating
16906a3 Thu Nov 12 18:30:13 2015 -0500 Alex Kalicki : use sprintf instead of string concatenation
583ad8c Thu Nov 12 18:22:12 2015 -0500 Alex Kalicki : minor: change order of compiler flag for
consistency with ocamlc
bc4a0c9 Thu Nov 12 18:18:40 2015 -0500 Alex Kalicki : add pre-python generator for conditionals, will
pretty-print braces as spaces
```

```
8389662 Thu Nov 12 18:04:45 2015 -0500 Alex Kalicki : Merge branch 'master' into conditionals
202a8f8 Thu Nov 12 18:03:13 2015 -0500 Alex Kalicki : parse assign name as id instead of string,
reorder ast.mli for consistency
2b3ba79 Thu Nov 12 17:51:36 2015 -0500 Alex Kalicki : Parse function call identifiers as Ids instead of
strings
80a446d Thu Nov 12 17:20:36 2015 -0500 Alex Kalicki : add function declaration to parser tests
9bb69cf Thu Nov 12 13:51:47 2015 -0500 dannyech : (Re)added ~voodoo~ magic function delimiter.
187d5db Thu Nov 12 12:03:20 2015 -0500 Alex Kalicki : make NOT right associative
9d54769 Thu Nov 12 12:00:31 2015 -0500 Alex Kalicki : reorganize parser
ad06729 Thu Nov 12 11:53:04 2015 -0500 Alex Kalicki : Merge commit '97c25ad' into functions2
bcea649 Thu Nov 12 11:47:12 2015 -0500 Alex Kalicki : get conditionals 2 to working state
348dd18 Thu Nov 12 11:38:56 2015 -0500 Alex Kalicki : Merge pull request #41 from
alexandramedway/hello-world
ecbe994 Thu Nov 12 04:40:38 2015 -0500 Alex Kalicki : add explicit hello world test for demonstration
purposes
97c25ad Thu Nov 12 01:18:41 2015 -0500 Alexandra Medway : Finally fixed testing
b76e0fc Thu Nov 12 01:12:37 2015 -0500 Alexandra Medway : Merge branch 'functions' of
https://github.com/alexandramedway/plt into functions
7a78c4e Thu Nov 12 01:12:27 2015 -0500 Alexandra Medway : Tests are now working
5bcc6b8 Thu Nov 12 00:55:34 2015 -0500 dannyech : Changed RETURN and NOT to Non-associative
5f7370e Thu Nov 12 00:50:05 2015 -0500 Alexandra Medway : Merge branch 'functions' of
https://github.com/alexandramedway/plt into functions
16b57a0 Thu Nov 12 00:49:51 2015 -0500 Alexandra Medway : Cleaning up tests
a013fcd Thu Nov 12 00:43:22 2015 -0500 dannyech : Changed order of types in ast.
f516800 Thu Nov 12 00:33:25 2015 -0500 Alexandra Medway : No more shift/reduce conflicts
0483795 Thu Nov 12 00:17:51 2015 -0500 Alexandra Medway : Merge and initial stages of function creation
bfb9741 Wed Nov 11 22:14:50 2015 -0500 Alexandra Medway : Reorganizing the Parser
d3df975 Wed Nov 11 19:01:33 2015 -0500 Danny : Merge pull request #39 from
alexandramedway/assignment_and_scoping
34412aa Wed Nov 11 18:58:24 2015 -0500 dannyech : Fixing merge conflicts
20b72be Wed Nov 11 18:56:04 2015 -0500 dannyech : Fixed Merge Conflict
02f62d6 Wed Nov 11 18:51:07 2015 -0500 dannyech : Merge branch 'master' into assignment_and_scoping
fbf606c Wed Nov 11 18:42:21 2015 -0500 Alex Kalicki : Merge pull request #37 from
alexandramedway/statements
c577be0 Wed Nov 11 18:37:21 2015 -0500 Alex Kalicki : replace Expr with Do
4e4a82c Wed Nov 11 18:27:28 2015 -0500 Alex Kalicki : remove sequencing tokens
d0ea2db Wed Nov 11 18:26:08 2015 -0500 Alex Kalicki : replace semicolons with do, remove sequencing
26abd23 Wed Nov 11 17:54:24 2015 -0500 Alex Kalicki : Merge branch 'master' into conditionals
44f7191 Wed Nov 11 17:52:52 2015 -0500 Alex Kalicki : Merge pull request #33 from
alexandramedway/booleans
1f60f61 Wed Nov 11 16:24:43 2015 -0500 dannyech : Combined update_ss_id and get_ss_id to a single
function because eveytime you retrieve a unique identifier, you need to update the ss_counter to keep
the identifier unique!
572c5da Mon Nov 9 15:18:18 2015 -0500 Alex Kalicki : nit: add newline to end of file
a2a84d3 Mon Nov 9 15:15:16 2015 -0500 Alex Kalicki : replace set and state keywords with new statement
endings, sequencing op
f6daff6 Mon Nov 9 12:16:08 2015 -0500 dannyech : 1. Added Set to AST under stmt. 2. Added dummy Set
pattern matching under process_stmt.
0d3fd64 Mon Nov 9 11:55:11 2015 -0500 dannyech : removed env variable from process statement list for
now
8e7946d Mon Nov 9 11:54:42 2015 -0500 dannyech : removed env variable from State(expr) for now
bdd5f75 Mon Nov 9 11:53:48 2015 -0500 dannyech : Changed name of static scoping vars and funcs from
cur_pyname/get_pyname to ss_counter/get_ss_id
24962ab Mon Nov 9 11:40:07 2015 -0500 Alex Kalicki : tidy up parser tests
9924125 Mon Nov 9 11:36:16 2015 -0500 Alex Kalicki : add if then else to parser and parser tests
8648e3f Mon Nov 9 11:27:20 2015 -0500 Alex Kalicki : remove Id() from Call parser tests since using a
string instead
7dd3d5d Mon Nov 9 11:07:55 2015 -0500 Alex Kalicki : separate parser expr into multiple parts for
readability
f533d71 Mon Nov 9 11:01:33 2015 -0500 Alex Kalicki : Merge pull request #34 from
alexandramedway/compiler-call-test
8a33610 Mon Nov 9 02:02:31 2015 -0500 Alex Kalicki : change Equal to Eq for consistency
7639a6b Mon Nov 9 00:59:42 2015 -0500 Alex Kalicki : reorder ast.mli lines for consistency
e804106 Sun Nov 8 21:47:00 2015 -0500 Alex Kalicki : add skeleton id/call compiler test to match parser
ones
```

```
1fc3612 Sun Nov 8 21:23:43 2015 -0500 Alex Kalicki : add boolean literals and relational operators
2cb6192 Sun Nov 8 19:52:45 2015 -0500 Alex Kalicki : Merge pull request #27 from
alexandramedway/compiler-test
7b750f1 Sun Nov 8 19:33:37 2015 -0500 dannyech : changed py_name so that the name of the odds
identifier can be passed to it. get_pyname is guaranteed to return a unique name because it appends the
current pyname reference to the end of the name passed in as an argument. Because we can pass a name by
argument, it will be easy to debug generated python code - the name's in it and their corresponding
odds variables will be similar, the only difference being the number at the end...
cae35e5 Sun Nov 8 19:23:54 2015 -0500 dannyech : updated get_pyname to account for the fact that python
variable names can have numbers in them
351772e Sun Nov 8 15:37:12 2015 -0500 dannyech : Added Set to Parser, added ground work for static
scoping
12fb5a2 Sun Nov 8 15:22:07 2015 -0500 lillyfwang : Merge branch 'compiler-test' of
https://github.com/alexandramedway/plt into compiler-test
ae1e69c Sun Nov 8 15:21:51 2015 -0500 lillyfwang : added literal tests for compiler
26a6267 Sun Nov 8 15:21:33 2015 -0500 Alex Kalicki : fix mktemp usage
604f4f2 Sun Nov 8 15:14:32 2015 -0500 Alex Kalicki : add compiler test infrastructure to make test
646951b Sun Nov 8 15:12:40 2015 -0500 Alex Kalicki : fix arithmetic compiler test files
ffe10d0 Sun Nov 8 15:05:19 2015 -0500 Alex Kalicki : Merge remote-tracking branch
'origin/compiler-tests' into compiler-test
0617b80 Sun Nov 8 15:04:52 2015 -0500 Alex Kalicki : add compiler test shell script
5fbbab9 Sun Nov 8 15:04:46 2015 -0500 lillyfwang : added compiler tests
c2d1d85 Sun Nov 8 15:04:38 2015 -0500 Alex Kalicki : allow odds shell script to be called from
different directories
03fb720 Sun Nov 8 14:01:08 2015 -0500 Alexandra Medway : Merge pull request #25 from
alexandramedway/compiler-script
b38e1ad Sun Nov 8 13:52:12 2015 -0500 Alex Kalicki : merge master into compiler-script
58ebf50 Sun Nov 8 13:50:09 2015 -0500 Alex Kalicki : Merge pull request #26 from
alexandramedway/parser-test-updates
cfb0efe Sun Nov 8 12:48:27 2015 -0500 dannyech : Changed name of compile.sh to odds.sh
089cc10 Sat Nov 7 16:25:39 2015 -0500 dannyech : put parens around unop to python text to guarantee
that the unary operator is tightly bound to the expression it is operating upon.
ca7a3f0 Sat Nov 7 16:19:34 2015 -0500 Alex Kalicki : add test for function call with no args
f12c582 Sat Nov 7 16:11:46 2015 -0500 Alex Kalicki : fixed parentheses precedence issue in generator
1521df3 Sat Nov 7 15:45:40 2015 -0500 Alex Kalicki : add test for precedence within functions (passing
test, not working in generator)
c0fefd2 Sat Nov 7 15:37:27 2015 -0500 Alex Kalicki : add tests for Ids, Unops, and function Calls
c575385 Sat Nov 7 15:01:22 2015 -0500 Alex Kalicki : move literals down in parser for better separation
of terminals and non
36b937f Sat Nov 7 14:55:19 2015 -0500 Alex Kalicki : minor changes for clarity
93227c3 Sat Nov 7 13:49:49 2015 -0500 Alex Kalicki : minor Makefile changes
f2d9c74 Sat Nov 7 13:46:28 2015 -0500 Alex Kalicki : merge branch master into compiler-script
710bda7 Sat Nov 7 13:42:08 2015 -0500 Alexandra Medway : Merge pull request #24 from
alexandramedway/compiler-basic
3993bab Sat Nov 7 13:30:18 2015 -0500 Alexandra Medway : Updated parser tests
f24a59b Sat Nov 7 13:25:24 2015 -0500 Alexandra Medway : Fixed bug, didn't have call
4513f54 Sat Nov 7 13:16:17 2015 -0500 Alexandra Medway : Merging
6e16fac Sat Nov 7 13:09:35 2015 -0500 Alexandra Medway : Nits, removing sast
1876ccb Fri Nov 6 19:42:30 2015 -0500 Alexandra Medway : Compile script complete. See compile.sh for
usage
f2fb30c Fri Nov 6 18:40:37 2015 -0500 Alexandra Medway : Starting the odds.ml script
35e8468 Fri Nov 6 16:17:48 2015 -0500 dannyech : changed process_stmt_list to use lists instead of
appending strings on each recursion
4b49095 Fri Nov 6 14:49:58 2015 -0500 dannyech : Reversed order of new string statement string
concatenation - now program does not print backwards
a9981c0 Fri Nov 6 14:28:41 2015 -0500 dannyech : Updated makefile to account for generator.ml's new
dependencies
b4c30d6 Fri Nov 6 14:01:05 2015 -0500 dannyech : generator.ml now outputs the program string rather
than printing sequentially to file. Added short (temporary) script to the end of generator.ml to
actually do the compilation.
bca06e1 Fri Nov 6 13:07:44 2015 -0500 dannyech : updated process_stmt
f6cd570 Fri Nov 6 13:03:34 2015 -0500 dannyech : update txt_of_func to txt_of_func_call. Added Call to
txt_of_Expr. Updated txt_of_args
ac0f5f2 Thu Nov 5 21:41:19 2015 -0500 dannyech : Changed txt_of_exp to txt_of_expr for sake of
consistency.
```

```
4c32666 Thu Nov 5 21:33:31 2015 -0500 dannyech : renamed *_to_text functions in generator.ml to
txt_of_* to fit with ocaml paradigms
ae1550f Thu Nov 5 20:19:55 2015 -0500 Alexandra Medway : Adding functions to parser, adding same to ast
1506792 Thu Nov 5 19:17:38 2015 -0500 Alex Kalicki : fix travis build image after repo rename
0349100 Thu Nov 5 14:16:26 2015 -0500 Alexandra Medway : Removed all syntax errors in generator. They
hadn't been raised before because the functions hadn't been tested directly
dbd1c61 Thu Nov 5 13:55:38 2015 -0500 Alexandra Medway : Changed Expr to State, tests are now passing
bae96ad Thu Nov 5 13:43:27 2015 -0500 Alexandra Medway : Merged parser with master
73df38a Thu Nov 5 13:41:24 2015 -0500 Alexandra Medway : Merged with master
52ce03c Thu Nov 5 12:17:52 2015 -0500 Alex Kalicki : remove scanner and parser Makefile targets since
we only need to build their objects
8fcb9c7 Thu Nov 5 01:46:37 2015 -0500 Alex Kalicki : playing god, adding back native compiler
dependency
4f19352 Thu Nov 5 01:41:53 2015 -0500 Alex Kalicki : Merge pull request #23 from
alexandramedway/test-performance
94c509b Thu Nov 5 01:21:00 2015 -0500 Alex Kalicki : remove annoying emails
12ebdf8 Thu Nov 5 01:14:33 2015 -0500 Alex Kalicki : attempt to remove ocaml-native-compilers
dependency
f5725e1 Thu Nov 5 01:11:31 2015 -0500 Alex Kalicki : Merge pull request #22 from
alexandramedway/parser-tests
198db21 Thu Nov 5 01:09:59 2015 -0500 Alex Kalicki : attempt to remove camlp4-extra dependency
a342177 Thu Nov 5 01:07:26 2015 -0500 Alex Kalicki : remove opam install in attempt to speed up build
performance
94eb5ee Thu Nov 5 01:04:11 2015 -0500 Alex Kalicki : add some space for readability
933ec6b Thu Nov 5 01:01:26 2015 -0500 Alex Kalicki : add test for parentheses
aa81a0a Thu Nov 5 00:53:16 2015 -0500 Alex Kalicki : add newline at end of file
467c8fe Thu Nov 5 00:34:01 2015 -0500 lillyfwang : added arithmetic tests for parser
3794d1b Thu Nov 5 00:05:58 2015 -0500 Alex Kalicki : Merge pull request #21 from
alexandramedway/parser-test-runner
227adda Thu Nov 5 00:04:01 2015 -0500 Alex Kalicki : nit: add newline to end of file
eebe744 Thu Nov 5 00:02:32 2015 -0500 Alex Kalicki : add newline to end of test output file
0baa85e Wed Nov 4 23:59:50 2015 -0500 Alex Kalicki : fix test and branch
e46fe7c Wed Nov 4 23:57:57 2015 -0500 Alex Kalicki : test breaking parser test workflow
fd6a66b Wed Nov 4 23:39:18 2015 -0500 Alex Kalicki : finish parser test infrastructure, add first tests
57c3a68 Wed Nov 4 23:01:02 2015 -0500 lillyfwang : uncommented code in scanner.mll
c89315c Wed Nov 4 22:57:34 2015 -0500 lillyfwang : added tokens to parser (deprecated tokens.mli),
deleted tokens.mli
8ca851e Wed Nov 4 22:46:54 2015 -0500 lillyfwang : fixed alex's dumb mistake
4e71cc3 Wed Nov 4 22:21:31 2015 -0500 Alex Kalicki : add initial parserizer
5f5d263 Wed Nov 4 22:19:16 2015 -0500 Alex Kalicki : work on part of test infrastructure
9e1a125 Wed Nov 4 22:15:45 2015 -0500 dannyech : added exp_to_text to generator.ml and then realized
all of this was for naught because we ought to do all the conversions to text in the sast file
fcbb6c2 Wed Nov 4 21:49:26 2015 -0500 dannyech : Preliminary code generation
222d61e Wed Nov 4 21:12:24 2015 -0500 Alexandra Medway : Created func_to_text and op_to_text
080b3db Wed Nov 4 20:26:57 2015 -0500 Alex Kalicki : Merge branch 'parser-test' of
github.com:alexandramedway/plt into parser-test-runner
b16fbf1 Wed Nov 4 20:26:39 2015 -0500 lillyfwang : created parserize tester (cst to string)
db7b12f Wed Nov 4 20:16:38 2015 -0500 Alexandra Medway : Merge branch 'parser-basic' of
https://github.com/alexandramedway/plt into compiler-basic
fcc8762 Wed Nov 4 20:11:36 2015 -0500 Alex Kalicki : Merge branch 'master' into parser-test-runner
33cbf76 Wed Nov 4 20:11:03 2015 -0500 Alexandra Medway : Adding initial declarations to the sast.mli
885c2d7 Wed Nov 4 20:09:50 2015 -0500 dannyech : Added Float_lit, String_lit, and Unop (for negatives)
e737510 Wed Nov 4 20:09:50 2015 -0500 Alexandra Medway : Added generator basics and sast basics
1f66e78 Wed Nov 4 19:13:26 2015 -0500 dannyech : Added FLOAT_LIT
e91b68a Wed Nov 4 18:56:40 2015 -0500 Alex Kalicki : Merge pull request #20 from
alexandramedway/test_rename
f72e935 Wed Nov 4 17:42:38 2015 -0500 Alex Kalicki : only allow statements that begin with 'state'
keyword
a976c86 Wed Nov 4 17:39:27 2015 -0500 Alex Kalicki : add eof token to parser, allow lists of
statements. works with menhir
5c51bc5 Wed Nov 4 17:26:19 2015 -0500 Alex Kalicki : add basic arithmetic parser, pre-testing
28fba90 Wed Nov 4 17:07:36 2015 -0500 Alex Kalicki : begin abstract syntax tree
649ea6e Wed Nov 4 16:46:48 2015 -0500 Alex Kalicki : Merge branch 'master' into test_rename
c78277a Wed Nov 4 16:28:02 2015 -0500 Alex Kalicki : renamed scanner test files for brevity
6f32a07 Sun Oct 25 23:07:13 2015 -0400 Lilly Wang : Merge pull request #14 from
```

```
alexandramedway/void-keyword
b1ffb2e Sun Oct 25 22:23:02 2015 -0400 lillyfwang : removed string with void literal
c43c683 Sun Oct 25 22:05:57 2015 -0400 lillyfwang : added void literal, removed colon from punctuation
5a46ade Wed Oct 21 16:17:42 2015 -0400 Alex Kalicki : Merge pull request #11 from
alexandramedway/log_cleanup
6342bad Wed Oct 21 01:17:30 2015 -0400 Alex Kalicki : clean up test log output, add colors :)
cbf2595 Mon Oct 19 18:21:00 2015 -0400 Alex Kalicki : Merge pull request #10 from
alexandramedway/scanner_test_func
0d8a25d Mon Oct 19 18:15:45 2015 -0400 Alex Kalicki : Merge remote-tracking branch 'origin/master' into
scanner_test_func
d29d4f3 Mon Oct 19 18:14:50 2015 -0400 Alex Kalicki : Merge pull request #7 from
alexandramedway/scanner_fixes
a44d42c Mon Oct 19 18:12:32 2015 -0400 Alexandra Medway : Removed binary files
004f8ed Mon Oct 19 18:09:30 2015 -0400 Alexandra Medway : Merge pull request #9 from
alexandramedway/additional_tests
2230390 Mon Oct 19 18:06:52 2015 -0400 Alexandra Medway : Merging
61f584e Mon Oct 19 18:00:54 2015 -0400 dannyech : Added Power token to tokenizer and ** to scanner.
Added ** to arithmetic tests. Added _mixed_arithmetic_literal test. fixed float_lit reggex. Closed #8,
Closed #2.
46bd765 Mon Oct 19 17:55:28 2015 -0400 Alexandra Medway : Merge branch 'scanner_fixes' of
https://github.com/alexandramedway/plt into additional_tests
e826708 Mon Oct 19 17:52:46 2015 -0400 Alexandra Medway : Added a test for comments
b012215 Mon Oct 19 16:50:33 2015 -0400 Alex Kalicki : Merge remote-tracking branch
'origin/scanner_fixes' into scanner_test_func
5d4a0ea Mon Oct 19 16:46:32 2015 -0400 Alex Kalicki : add more complicated scanner test case for iseven
function
51a49ca Mon Oct 19 16:40:57 2015 -0400 dannyech : Removed unnecessary character set delimiters from
float regex
9297f05 Mon Oct 19 16:39:22 2015 -0400 dannyech : Added '-'? to float regex to recognize negative
floats
5ff2276 Mon Oct 19 16:32:27 2015 -0400 dannyech : fixed 2 issues with scanner (details in description
section)
f3c1462 Mon Oct 19 16:16:53 2015 -0400 Alex Kalicki : Merge pull request #4 from
alexandramedway/test_framework
94948ac Mon Oct 19 16:16:31 2015 -0400 Alex Kalicki : add travis email notifications for build status
change
5e5648f Mon Oct 19 15:48:54 2015 -0400 Lilly Wang : Merge pull request #5 from
alexandramedway/scanner_tests
1bcaccc Mon Oct 19 15:46:04 2015 -0400 lillyfwang : Updated test for identifiers to include one that
starts with an underscore
56c17ab Mon Oct 19 15:41:01 2015 -0400 lillyfwang : Merge remote-tracking branch
'origin/test_framework' into scanner_tests
93240d2 Mon Oct 19 15:32:13 2015 -0400 lillyfwang : Added tests for all the different types of tokens
a03eb4c Mon Oct 19 15:30:20 2015 -0400 Alex Kalicki : remove bad test files
eb122de Mon Oct 19 15:26:38 2015 -0400 Alex Kalicki : check correct behavior on test fail
48f7732 Mon Oct 19 15:21:19 2015 -0400 Alex Kalicki : try bash instead of sh with travis
93c888c Mon Oct 19 15:16:46 2015 -0400 Alex Kalicki : no dependencies to install
1d53114 Mon Oct 19 15:12:50 2015 -0400 Alex Kalicki : enable sudo in attempt to fix travis
f8b4db9 Mon Oct 19 14:55:38 2015 -0400 Alex Kalicki : add 'make all' paths to each Makefile
f166916 Mon Oct 19 14:27:38 2015 -0400 Alexandra Medway : Added make all to the Makefiles
9d5f9f7 Mon Oct 19 14:19:57 2015 -0400 Alexandra Medway : Added dependencies to the Makefiles so that
when you type make all, instead of giving warning about things not being built in parent directories,
it makes the parent directories.
7cd9923 Mon Oct 19 13:46:06 2015 -0400 Alex Kalicki : add travis config
c0ee7c7 Mon Oct 19 13:44:49 2015 -0400 Alex Kalicki : #3 - add initial tests, hook bash file up to
tokenize
b690b75 Mon Oct 19 13:22:45 2015 -0400 Alex Kalicki : Merge branch 'test_scan_compare' into
test_framework
db9f5fa Mon Oct 19 13:22:26 2015 -0400 Alex Kalicki : Merge branch 'scan_print_test' into
test_framework
4632775 Mon Oct 19 13:21:01 2015 -0400 Alex Kalicki : work on #3 - resolve linking and create makefiles
88ad945 Mon Oct 19 12:52:16 2015 -0400 Alex Kalicki : fix stringify tokens
2f6a578 Mon Oct 19 12:49:19 2015 -0400 Alex Kalicki : move functions to correct order, remove dummy
input file
c2aa74f Mon Oct 19 00:25:15 2015 -0400 Lilly Wang : Delete scanner.mll (test scanner)
```

```
56f72f4 Mon Oct 19 00:19:44 2015 -0400 lillyfwang : Merge branch 'master' of
https://github.com/alexandramedway/plt
3ce24da Mon Oct 19 00:19:32 2015 -0400 lillyfwang : Added a .gitignore file
f11bf1e Mon Oct 19 00:15:39 2015 -0400 lillyfwang : Finished writing test for scanner, updated test
file name
1ac8ce8 Mon Oct 19 00:07:27 2015 -0400 lillyfwang : Added makefiles and test files
9dced63 Sun Oct 18 23:08:01 2015 -0400 Alex Kalicki : write error to stderr
6b9c78e Sun Oct 18 21:29:12 2015 -0400 Alex Kalicki : add travis ocaml config script
fb8811c Sun Oct 18 21:19:34 2015 -0400 Alex Kalicki : newline nits
078fdd4 Sun Oct 18 21:18:50 2015 -0400 Alex Kalicki : newline in scanner test
03857ee Sun Oct 18 21:16:24 2015 -0400 Alex Kalicki : final readme nits
d63eb80 Sun Oct 18 21:15:44 2015 -0400 Alex Kalicki : readme nits
59374ff Sun Oct 18 21:14:32 2015 -0400 Alex Kalicki : format names in readme
9136226 Sun Oct 18 21:13:50 2015 -0400 Alex Kalicki : initial makefile
ae58c87 Sun Oct 18 20:54:29 2015 -0400 Alex Kalicki : add initial travis file
e960ba7 Sun Oct 18 20:47:50 2015 -0400 Alex Kalicki : add ci image/link to README
331289a Sun Oct 18 20:38:40 2015 -0400 Alex Kalicki : Write initial scanner test script
855c6d9 Sun Oct 18 17:42:04 2015 -0400 Alex Kalicki : add spacing for consistency
29d0a51 Sun Oct 18 17:13:43 2015 -0400 Alexandra Medway : Merge pull request #1 from
alexandramedway/parser_and_scanner
9ca2f56 Sun Oct 18 17:08:34 2015 -0400 Alexandra Medway : Retabbing
a856d03 Sun Oct 18 17:03:04 2015 -0400 Alexandra Medway : Merge branch 'parser_and_scanner' of
https://github.com/alexandramedway/plt into parser_and_scanner
8d76f42 Sun Oct 18 17:02:33 2015 -0400 Alexandra Medway : Removed static typing, changed id name typing
123f1b1 Wed Oct 14 20:46:56 2015 -0400 dannyech : Edited string literal regular expression
f094a48 Wed Oct 14 20:40:48 2015 -0400 dannyech : Basic Functionality of Scanner Completed
0d0b558 Wed Oct 14 19:02:02 2015 -0400 dannyech : Initial scanner setup
7673e94 Wed Oct 14 18:33:18 2015 -0400 Alexandra Medway : Added Names To README
449d27d Wed Oct 14 18:17:04 2015 -0400 dannyech : Made parser file and scanner file
53a1fd9 Wed Sep 23 16:41:39 2015 -0400 Alexandra Medway : Final test
3944315 Wed Sep 23 16:40:59 2015 -0400 Alexandra Medway : Test
0c2fcfd Wed Sep 23 16:33:20 2015 -0400 Alexandra Medway : Testing the readme
fce2dc4 Wed Sep 23 16:29:24 2015 -0400 Alexandra Medway : Initial commit
```