The Setup Programming Language

Ian Erb (ire2102)
Bill Warner (whw2108)
Adam Weiss (ajw2137)
Andrew Ingraham (aci2110)

December 23, 2011

Contents

Chapter 1

Introduction

The SETUP language was created to leverage the clear and concise syntax of set formalisms in mathematics. With SETUP , users can generate sets using a variety of logic and combinatorial techniques that would require significantly more lines of code in other imperative languages such as Java, C or C++. SETUP was designed with the following goals in mind:

- **Set Theory Abstraction.** SETUP provides a framework for handling data which many mathematicians and scientists will find familiar and easy to use. By maintaining a sufficient level of abstraction, SETUP can minimize the time needed to go from concept to concrete, working code.
- **Minimal Code Generation.** SETUP greatly reduces the amount of code needed to perform routine tasks. The user will find that nested loops can be virtually eliminated from code, making program files lighter and debugging easier.
- **High Level of Readability.** SETUP aims to be a clear and concise programming language with intuitive commands and syntax which mirror the mathematical underpinnings of set theory.

Chapter 2

Setup Tutorial

Each program you write in SETUP should be placed in a plain text file. By convention, the `.su` suffix is used to denote programs written in valid SETUP code. Let's write a simple program to get started in SETUP .

2.1 A Simple Scheduling Program

In this tutorial we will write a simple SETUP program which demonstrates an application of the language's set abstraction in the manipulation to compute some probabilities relating to games of chance. In particular, we're going to compute probabilities relating to a hand of poker.

We begin by creating a plain text file to store our code. Let's give it the title `pokerhand.su` and open it in your text editor. In this file, place the following text

```
/* Our First Program */

function main[] returns int {
        set values = {1 ... 13};
        set suit = {"H","S","C","D"};
        set deck = values cross suit;
        set ourCards = {(2,"S"),(7,"S"),(8,"S"),(9,"S"),(10,"D")};
        set restOfDeck = deck minus ourCards;
        set test = values cross {"S"};
        set flushCards = test minus ourCards;
         float prob_of_flush = #flushCards / #restOfDeck;
        print(prob_of_flush);
        return 0;
        }
```

You can compile the program into C++ code by calling

```
-$ ./cWriter.native < pokerhand.su
```

Alternatively, you can call

```
-$ ./cWriter.sh < dice1.su
```

which will print, compile, and run the code immediately in the shell.

In this small program all functionality /*except for a comment which is ignored by the compiler */ was placed inside the definition of `main`. This is not always the case. User-defined functions and global variable declarations may be made outside of `main` in what is known as *global scope*. Global variable definitions may not contain initializers.

SETUP programs begin execution with `main`. In this program, we created a collection of "cards" by using the cartesian product keyword `cross`. You will also notice that `{1 ... 13}` expanded to include integers in the range $[1, 13]$. After declaring our cards using literals, we "removed" them from the deck using the set difference operator `minus`. After generating a set of possible flush cards, we used the unary "cardinality" operator to compute the probability of hitting a flush: 19%.

Chapter 3

Language Reference Manual

3.1 Overview

3.1.1 Introduction

The language of set theory is widely used by mathematicians and scientists to construct and manipulate large collections of objects in an abstract setting. The SETUP language implements the most useful and commonly used abstractions (e.g., union, intersections, direct products) from set theory in a language which users of other common imperative languages such as C, C++ and Java will find very simple and intuitive.

SETUP was created with three goals in mind:

1. Construct sets in a logical way
2. Perform operations on large sets with minimal code generation
3. Maintain a high level of readability

We anticipate users will solve simple set-oriented problems like scheduling, logic, and probability problems.

3.1.2 A Motivating Example

SETUP greatly reduces the amount of code required to generate and manipulate sets. A level of abstraction is provided which allows the user to work with sets in an intuitive way – simultaneously improving readability. For example, the need for writing successive nested loops is practically eliminated.

Suppose the user wanted to generate all possible ordered pairs of numbers from 1 to 10. In C++, we would write:

```
struct { int first; int second; } pair;
vector<pair> pairs;
for(int i = 1; i!= 11; i++){
    for(int j = 1; j!=11; j++){
                pair temp;
                temp.first = i;
                temp.second = j;
        vector.push_back(temp);
    }
}
```

In SETUP , the equivalent code can be written:

```
set A = {1...10};
set pairs = {(x,y) | x in A, y in A};
```

Making use of `cross` – a built-in cartesian product operator – we can further simplify our code to a single line:

```
set pairs = {1...10} cross {1...10};
```

Though somewhat trivial, the above example shows how we can greatly reduce code and improve readability at the same time. Suppose we wanted to remove duplicates `(1,1) ... (10,10)`. We would simply write

```
set unique_pairs = pairs minus {(x,x) | x in {1...10}};
```

Here we highlight another useful built in operator – `minus` – which returns the complement of the second set within the first set. That is, `A minus B` $= A \cap B^c$.

Working with SETUP is simple and getting started is easy.

3.1.3 Lexical Conventions

The language has 5 basic tokens:

- Identifiers
- Keywords
- Literals
- Operators
- Punctuation

Whitespace characters (blanks, tabs and newlines) are ignored and used only to separate tokens. At least one whitespace character is required to separate adjacent tokens.

Comments

Block comments are introduced with `/*` and terminated with `*/`. Nesting of comments is not permitted. Comments are in general ignored by the compiler.

Identifiers

An identifier is a sequence of letters and digits. The first character must be alphabetic. Names are case-sensitive.

Keywords

The following identifiers are reserved for use as keywords, and may not be used otherwise:

`set`	`int`	`bool`	`float`	`string`
`if`	`tuple`	`in`	`then`	`union`
`else`	`intersect`	`minus`	`while`	`cross`
`true`	`function`	`returns`	`false`	`return`

Primitive Types

There are four primitive types in SETUP :

Integers

An integer is a sequence of digits. All integers are lexed as a sequence of digits with an optional leading minus sign for negative integers. They are represented internally using architecture native integer representation.

Strings

A string is a sequence of characters enclosed in double quotes as in `"string"`. Two adjacent strings are concatenated using the "+" sign. As in

```
"string" + "concat" -> "stringconcat".
```

Floats

We adopt the *C Reference Manual* definition of a floating point number:

> A floating constant consists of an integer part, a decimal point, a fraction part, an `e` and an optionally signed integer exponent. The integer and fraction parts both consist of a sequence of digits. Either the integer part or the fraction part (not both) may be missing; either the decimal point or the `e` and the exponent (not both) may be missing.

All floating point numbers will be 64-bit double precision.

Booleans

A Boolean value can take either `true` or `false`.

3.2 Syntax Notation

In this manual, elements of language syntax are indicated by *italic* type. Literal words and characters are written in `verbatim`. Alternatives are listed using the "|" character: *item* | *item*.

3.3 Objects

3.3.1 Variables

A variable token is an identifier to a stored primitive, tuple or set. A variable must begin with an alphabetic character followed by zero or more letters and digits. Variables are declared by referencing their type followed by the associated token and an optional initializer, as in `int a;` or `int a = 3;`. All global variables must be declared outside of any function definition, and cannot include an initializer.

A variable, once declared, may be reassigned but cannot be declared again in the same scope.

```
int a = 3;
a=4;         /* ok    */
int a = 5;   /* error */
```

Variable Initialization

Variables for primitives not explicitly initialized when declared will be initialized as follows:

$$int \rightarrow 0$$
$$string \rightarrow \text{""}$$
$$float \rightarrow 0.0$$
$$bool \rightarrow false$$
$$set \rightarrow \{\}$$

Uninitialized tuples are not permitted.

3.3.2 Tuples

A *n-tuple* is an ordered collection of n comma-delimited *expressions* enclosed in parentheses. An *element* is either a primitive, set or tuple. An *n-tuple* and *m-tuple* are considered of the same type if the following two conditions are satisfied:

- $n = m$.
- The type of each coordinate *element* is of the same type.

For example, the following tuple elements are not the same type because the first coordinate *tuples* are not of the same *type*:

```
((1,"a"),2) //type: ((int,str),int)
((1,2),3) //type: ((int,int),int)
```

3.3.3 Sets

A set is an unordered collection (potentially empty) of terminals, sets or tuples. Every element of a set is unique (duplicate elements are discarded). A set must be **homogeneous in type**, meaning that every element within the set has matching type. All sets are typed as `set`, regardless of their contents. This means that a set containing sets of varying types is allowed.

```
{1,2,3,4} /* valid: homogeneous in type */
{1,2,3,"f"} /* invalid: not homogeneous in type */
{(1,2),(3.0,4.0)} /* invalid: tuple types are deep */
{{1,2,3},{"a","b"}} /* valid: set types are shallow */
```

Sets can be initialized in various ways:

Literal Initialization

A set may be initialized with a comma-delimited list of elements or identifiers of matching type within curly braces:

```
set A = {1, 2, 3, 4, 5, 6}; /* ok */
string b = "name";
set B = {"this", "works", b}; /* ok */
set C = {1, 2, 3, b}; /* error: type mismatch */
```

Range Initialization

For sets of integer type, we allow the following range initialization using "`...`":

```
set A = {1 ... 6}; /* ok: returns {1, 2, 3, 4, 5, 6} */
set B = {1 ... 3, 5...7}; /* ok: returns {1, 2, 3, 5, 6, 7} */
set C = {3 ... -1}; /* ok: returns {3, 2, 1, 0, -1} */
```

Set-Builder Initialization

The following syntax is used for initializing a set through set-builder notation:

`{` *expr pipe sourcelist* `}`

The source list is a sequence of comma-delimited expressions of the form

id `in` *set*

Each *id* represents a local variable which may be used to construct new elements for a set via the expression appearing on the left hand side of the | symbol. The sequenced sourcelist expressions are evaluated left-to-right.

The `in` operator is right associative, and the *id* is not assigned a value until the right operand has been resolved. The resulting set will include the resulting left side expression evaluated for all potential *id* values, with duplicates removed.

```
set A = { (x,y) | x in {1,2,3}, y in {"a","b","c"} };
B = { (1,"a"),(1,"b"),(1,"c"),
      (2,"a"),(2,"b"),(2,"c"),
      (3,"a"),(3,"b"),(3,"c") };
A == B; /* returns yup */
```

In addition, the left side expression may contain references to variables in enclosing scopes (local sourcelist variables shadow enclosing scopes). For example:

```
int a = 0;
int x = 5;
set A = { (a,x) | x in {1,2,3} }; /*x shadows enclosing x = 5*/
B = { (0,1),(0,2),(0,3) };
A == B; /*returns yup*/
```

3.4 Operators

3.4.1 Arithmetic Operations

The following arithmetic operations are provided: `+,-,*` for types `int` and `float`. The return type is as follows:

$$\textit{int binop int} \rightarrow \textit{int}$$
$$\textit{float binop float} \rightarrow \textit{float}$$
$$\textit{float binop int} \rightarrow \textit{float}$$
$$\textit{int binop float} \rightarrow \textit{float}$$

There are two division operators:

1. `/`, which accepts as arguments any two numerical values and is guaranteed to return a `float`
2. `//` accepts as arguments any two numerical values and is guaranteed to return an `int` truncated toward zero.

The following relational operators are provided for all primitive types: `<,>,<=,>=,==,!=`. Types passed to these operators must match, except in the case of `<,>` when comparing `int` and `float`. String comparison is done lexicographically as per `strcmp` in the *C* standard library. The only character set supported is 8-bit ASCII.

3.4.2 Set Operations

The following set operations are provided:

`union  intersect  minus  cross  #`

union

`union` is a binary operator which returns a union of two sets. Both sets must contain elements of the same type.

intersect

`intersect` is a binary operator which returns the intersection of two sets. Both sets must contain elements of the same type.

minus

`minus` is a binary operator which returns a set of elements which are present in the first set but **not** in the second set. Both sets must contain elements of the same type.

cross

`cross` takes as left operand a set with elements of type a and as right operand a set of elements with type b and returns an exhaustive set of tuples of type (a, b), duplicates removed.

#

`#` is a unary operator returning the number of elements in a set.

3.5 Language Syntax

3.5.1 Structure of a Setup Program

Every SETUP progam is a sequence of zero or more function definitions and/or variable declarations. Variable declarations outside of function bodies are considered global in scope and may not be assigned a value when declared.

$$\begin{aligned} &\textit{program} \rightarrow \epsilon \\ &\textit{program} \rightarrow \textit{funcdef program} \\ &\textit{program} \rightarrow \textit{vdecl program} \end{aligned}$$

Execution (of non-empty programs) begins by calling the `main` function which returns an `int`. User-defined functions must be defined before the `main` function. Nested function definitions are not permitted.

3.5.2 Expressions

Terminals

$$\textit{expr} \rightarrow \textit{float} \mid \textit{int} \mid \textit{string} \mid \textit{bool} \mid \textit{id}$$

Tuples

expr → *tuple*
tuple → `(` *expr-list* `)`
expr-list → *expr* | *expr*`,`*expr-list*

Sets

expr → *set*
set → `{` *expr-list* `}`
set → `{` *int* `...` *int* `}`
set → `{` *expr* | *source-list* `}`
source-list → *id* `in` *expr* | *id* `in` *expr, source-list*

Operations

expr → *id* `=` *expr*
expr → *expr binop expr*
expr → *unop expr*

3.5.3 Functions

Function definitions begin with the keyword `function` followed by a valid identifier (see 3.1.3), a possibly empty list of formals, a return type specified using the keyword `return` and a curly-brace enclosed list of statements.

funcdef → `function` *id* `[` *formals-list* `]` `returns` *typespec* `{` *stmt-list* `}`
formals-list → ϵ | *formals-tail*
formals-tail → *formal* | *formal formals-tail*
formal → *typespec id*
typespec → `int` | `set` | `float` | `string` | `tuple` | `bool`

Function definitions specify the expression to be returned using a return statement:

`return` *expr*`;`

Functions are called by specifying an *id* and a list of arguments enclosed in parentheses:

funcall → *id*`(` *arglist* `)`
arglist → ϵ | *exprlist*
exprlist → *expr* | *expr* `,` *exprlist*

3.5.4 Statements

A statement list is simply a sequence of one or more statements:

$$\textit{stmt-list} \rightarrow \textit{stmt} \quad | \quad \textit{stmt stmt-list}$$

with the following valid statements:

Variable Declaration and Assignment

$$\textit{stmt} \rightarrow \textit{typespec id};$$
$$\textit{stmt} \rightarrow \textit{typespec id} \texttt{ = } \textit{expr};$$

Control Statements

$$\textit{stmt} \rightarrow \texttt{if(}\ \textit{expr}\ \texttt{) then \{}\ \textit{stmt-list}\ \texttt{\} else \{}\ \textit{stmt-list}\ \texttt{\}}$$
$$\textit{stmt} \rightarrow \texttt{while (}\ \textit{expr}\ \texttt{) \{}\ \textit{stmt-list}\ \texttt{\}}$$
$$\textit{stmt} \rightarrow \texttt{return (}\ \textit{expr}\ \texttt{);}$$

Definitions

Print Statement

SETUP provides a method for nice printing to the standard output:

$$\textit{stmt} \rightarrow \texttt{Print (}\ \textit{expr}\ \texttt{);}$$

Expressions

A statement may also be an expression:

$$\textit{stmt} \rightarrow \textit{expr};$$

3.6 Scope Rules

There exists a global scope containing global variables and function definitions. Each function body and set-builder expression defines a local scope. Variable identifiers without type definitions are treated as references; local scope is first searched, followed by successive enclosing scopes. Variable identifiers with type definitions are treated as new declarations, and mask all variable identifiers in enclosing scopes with the same identifier.

Sourcelist variables exist in the left side of set-bulider notation:

{expr pipe sourcelist}

A simple example of set-builder scope:

```
set A1 = {1,2};
set A2 = {3,4};
set A = {A1,A2};

/* result: B = {(1,2),(2,1),(3,4),(4,3)} */
set B = {(x,y) | a in A, x in a, y in a-{x}};
```

A richer example, using nested scopes:

```
set A1 = {1,2};
set A2 = {3,4};
set A = {A1,A2};

/* result: B = {4,6,8,10} */
set B = {x*2 | a in A, x in {y+1 | y in a } };

          /*Pseudocode:
        foreach a in A
            foreach y in a
                x = y+1
                B.add(x*2)
            next y
        next a
           */
```

An example of variable shadowing:

```
int x = 2;

/* result: A = {4,16} */
set A = {x*x | x in {y | y in {x,x*x} } };
x; /* result: x=2 */

      /*Pseudocode
        x = 2
        temp t = {x, x*x}
        foreach y in t
            x=y;
            A.add(x*x)
        next y
       */
```

Chapter 4

Project Plan

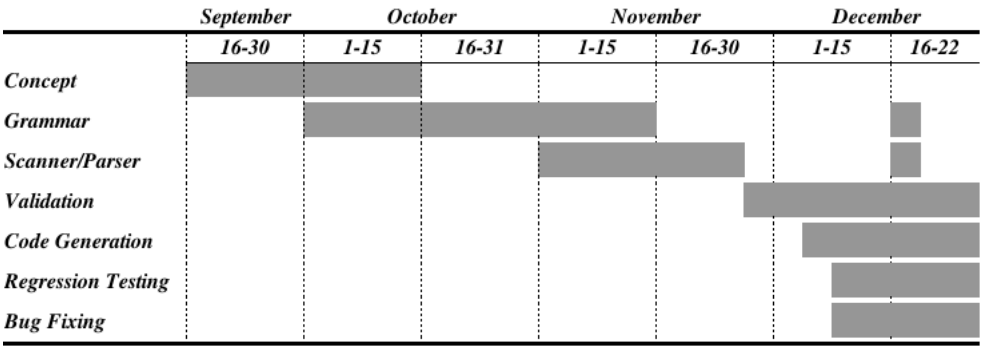
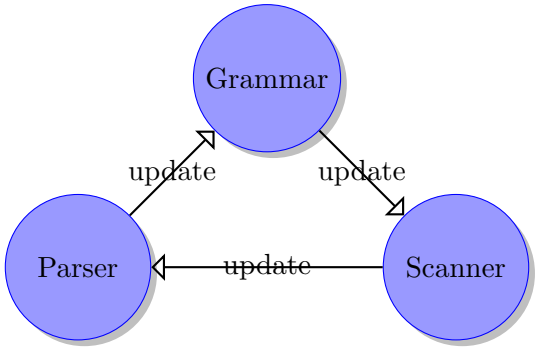


Figure 4.1: Timeline for Setup Team

September. Our team held weekly meetings to discuss different potential ideas for a language. By the end of the month, we had decided on a language that would manipulate sets. We then set to work developing a rough draft for a working grammar and began drafting our reference manual.

October. In conjunction with writing the first draft of our LRM, we began work on the scanner and parser. As we built the parser, we found several areas where our grammar needed to be "improved." The following graphic illustrates nicely how we spent October:



November. We completed work on our parser in November and began working on validation (static semantic analysis) to perform type checking. By the end of the month we were at the point where we could start generating code.

December. Writing code to generate valid C++ turned out to be more difficult than we anticipated, largely due to deep typing of nested tuples, and we continued to struggle with the Set-Builder construction mechanism. We implemented a set class abstraction in C++ to handle some of the peculiarities of arbitrary set and tuple containers. Throughout December, we generated code and tested for bugs.

Software Used. All members of our team used Vim or Vi as our editor of choice and we used GitHub for version control to stay synchronized in our development.

Chapter 5

Compiler Architecture

The compiler consists of 5 principal components: Scanner, Parser, Validator, Code Generator and Set Class. The Scanner receives input in the form of plain text files with `.su` suffix and the compiler generates valid cpp code.

The Code Generator uses the concrete parse tree generated from the Parser to output valid C++ code. A polymorphic class hierarchy was implemented in C++ (Set Class component) to implement complicated Tuple and Set types.

The following lays out who worked on the various component implementations:

Component	**Team Members**
Scanner	Adam
Parser	Adam, Ian
Validation	Bill
Code Generator	Andrew, Bill, Ian
Set Class	Bill, Andrew
Regression Suite	Ian, Bill, Andrew

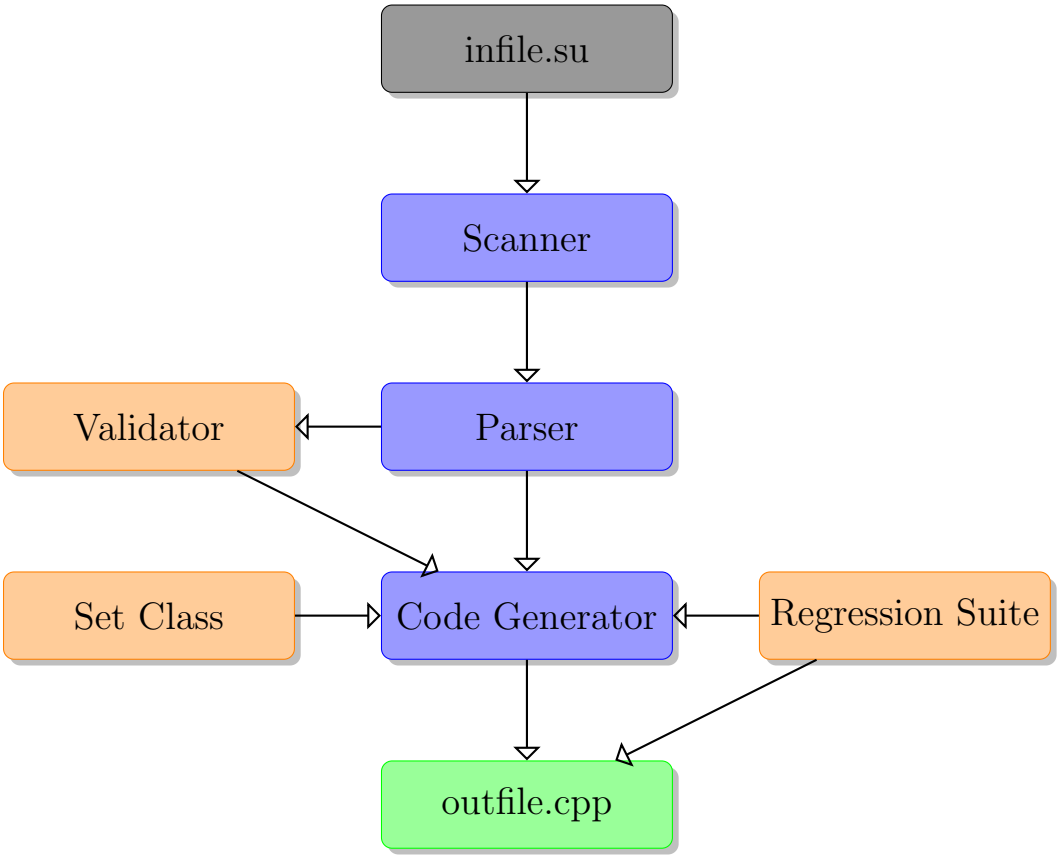


Figure 5.1: Overview of Setup Compiler

Chapter 6

Test Plan

We maintain a working regression suite which tests the various parts of our compiler. Over 120 tests were used. Python Scripts were used for automation. Tests focused on most common errors (arithemtic, assignment, etc.) and ensured accurate results by testing against golden `.su.out` files.

6.1 Parser Tests

```
 -LINES-       --FILENAME--
    5 ./tests/parser/expect_failure/fail2.su
    4 ./tests/parser/expect_failure/fail3.su
    5 ./tests/parser/expect_failure/fail1.su
    5 ./tests/parser/expect_success/comment1.su
   30 ./tests/parser/expect_success/succeed1.su
   11 ./tests/parser/expect_success/global.su
    4 ./tests/parser/expect_success/float1.su
    5 ./tests/parser/expect_success/initial.su
\
```

6.2 Code Generation Tests

```
 -LINES-       --FILENAME--
 10 ./tests/cWriter/expect_failure/string.su
   10 ./tests/cWriter/expect_success/string.su
   11 ./tests/cWriter/expect_success/add1.su
    1 ./tests/cWriter/expect_success/set_range.su
    6 ./tests/cWriter/expect_success/test1.su
    1 ./tests/cWriter/expect_success/simplest_set.su
```

```
 9 ./tests/cWriter/expect_success/comments.su
25 ./tests/cWriter/expect_success/arithmetic_int_retvals.su
 8 ./tests/cWriter/expect_success/test3.su
19 ./tests/cWriter/expect_success/arithmetic_float_vars.su
 8 ./tests/cWriter/expect_success/arithmetic_float_literals.su
21 ./tests/cWriter/expect_success/arithmetic_int_vars.su
11 ./tests/cWriter/expect_success/test2.su
13 ./tests/cWriter/expect_success/cross.su
29 ./tests/cWriter/expect_success/add.su
 5 ./tests/cWriter/expect_success/printTest1.su
 8 ./tests/cWriter/expect_success/ndivide.su
 1 ./tests/cWriter/expect_success/nested_set.su
 9 ./tests/cWriter/expect_success/test12.su
13 ./tests/cWriter/expect_success/function_return_set.su
 8 ./tests/cWriter/expect_success/conversion.su
11 ./tests/cWriter/expect_success/if_then.su
13 ./tests/cWriter/expect_success/union.su
 6 ./tests/cWriter/expect_success/minus.su
 1 ./tests/cWriter/expect_success/nested_set1.su
13 ./tests/cWriter/expect_success/arithmetic_float_retvals.su
 8 ./tests/cWriter/expect_success/arithmetic_int_literals.su
11 ./tests/codeGenTests/expect_success/test11.su
 7 ./tests/codeGenTests/expect_success/test7.su
10 ./tests/codeGenTests/expect_success/test4.su
 6 ./tests/codeGenTests/expect_success/test1.su
 6 ./tests/codeGenTests/expect_success/test8.su
 5 ./tests/codeGenTests/expect_success/test3.su
 9 ./tests/codeGenTests/expect_success/test2.su
 5 ./tests/codeGenTests/expect_success/test9.su
 7 ./tests/codeGenTests/expect_success/test5.su
 5 ./tests/codeGenTests/expect_success/test10.su
 9 ./tests/codeGenTests/expect_success/test6.su
 7 ./genTests/failing/tuple3.su
 9 ./genTests/failing/paren2.su
17 ./genTests/failing/rec_func1.su
10 ./genTests/failing/global2.su
14 ./genTests/failing/func_ret2.su
16 ./genTests/failing/paren3.su
 8 ./genTests/failing/tuple2.su
27 ./genTests/failing/rec_func2.su
10 ./genTests/failing/paren1.su
10 ./genTests/failing/func_ret6.su
10 ./genTests/failing/func_ret7.su
```

```
  13 ./genTests/succeeding/func_ret3.su
  23 ./genTests/succeeding/whileloop.su
  14 ./genTests/succeeding/func_ret5.su
  10 ./genTests/succeeding/global1.su
  14 ./genTests/succeeding/func_ret1.su
  14 ./genTests/succeeding/func_ret4.su
  20 ./demo/database.su
  57 ./demo/setOp.su
   9 ./demo/fib.su
   5 ./demo/poker.su
  21 ./demo/setDiff.su
```

6.3 Type-Checking and Validation

```
-LINES-       --FILENAME--
   5 ./tests/validation/expect_failure/logor_int.su
   7 ./tests/validation/expect_failure/setbuilder.su
   5 ./tests/validation/expect_failure/minus_string.su
   5 ./tests/validation/expect_failure/mult_string.su
   7 ./tests/validation/expect_failure/while.su
   5 ./tests/validation/expect_failure/compare11.su
   6 ./tests/validation/expect_failure/two_decl.su
   6 ./tests/validation/expect_failure/float2.su
   5 ./tests/validation/expect_failure/tuple3.su
   5 ./tests/validation/expect_failure/compare8.su
   5 ./tests/validation/expect_failure/logand_string.su
   4 ./tests/validation/expect_failure/return_type.su
   7 ./tests/validation/expect_failure/setrange.su
   5 ./tests/validation/expect_failure/undeclared.su
  11 ./tests/validation/expect_failure/func_mutual_recursion.su
   6 ./tests/validation/expect_failure/plus_mix.su
   6 ./tests/validation/expect_failure/float1.su
   5 ./tests/validation/expect_failure/decl_clash.su
   5 ./tests/validation/expect_failure/compare4.su
   5 ./tests/validation/expect_failure/plus_set.su
  12 ./tests/validation/expect_failure/return_type_deeper.su
   5 ./tests/validation/expect_failure/compare2.su
   5 ./tests/validation/expect_failure/compare5.su
   5 ./tests/validation/expect_failure/compare7.su
   5 ./tests/validation/expect_failure/tuple4.su
   5 ./tests/validation/expect_failure/set_literal.su
   5 ./tests/validation/expect_failure/minus.su
```

```
 5 ./tests/validation/expect_failure/tuple.su
 5 ./tests/validation/expect_failure/compare1.su
 5 ./tests/validation/expect_failure/compare10.su
 5 ./tests/validation/expect_success/compare_string.su
 8 ./tests/validation/expect_success/setbuilder.su
19 ./tests/validation/expect_success/setbuilder1.su
 5 ./tests/validation/expect_success/op_mix1.su
17 ./tests/validation/expect_success/setbuilder4.su
 7 ./tests/validation/expect_success/while.su
 4 ./tests/validation/expect_success/hrm_why.su
 5 ./tests/validation/expect_success/op_mix3.su
 5 ./tests/validation/expect_success/op_mix7.su
41 ./tests/validation/expect_success/func_scope8.su
 5 ./tests/validation/expect_success/op_mix8.su
 5 ./tests/validation/expect_success/tuple3.su
 9 ./tests/validation/expect_success/return_type.su
 5 ./tests/validation/expect_success/compare_float.su
11 ./tests/validation/expect_success/func_scope2.su
 5 ./tests/validation/expect_success/setrange.su
16 ./tests/validation/expect_success/setbuilder2.su
 5 ./tests/validation/expect_success/simple_plus.su
29 ./tests/validation/expect_success/add.su
 6 ./tests/validation/expect_success/float1.su
 7 ./tests/validation/expect_success/setrange2.su
10 ./tests/validation/expect_success/if_simple.su
 5 ./tests/validation/expect_success/logor.su
 5 ./tests/validation/expect_success/tuple2.su
13 ./tests/validation/expect_success/return_type_deeper.su
 5 ./tests/validation/expect_success/string_plus.su
11 ./tests/validation/expect_success/func_scope.su
18 ./tests/validation/expect_success/if.su
 5 ./tests/validation/expect_success/op_mix6.su
 5 ./tests/validation/expect_success/op_mix5.su
 5 ./tests/validation/expect_success/logand.su
 8 ./tests/validation/expect_success/tuple4.su
 5 ./tests/validation/expect_success/op_mix2.su
 5 ./tests/validation/expect_success/set_literal.su
 5 ./tests/validation/expect_success/minus.su
 5 ./tests/validation/expect_success/op_mix4.su
 5 ./tests/validation/expect_success/tuple.su
10 ./tests/validation/expect_success/func_scope4.su
 5 ./tests/validation/expect_success/compare.su
17 ./tests/validation/expect_success/setbuilder3.su
```

Chapter 7

Lessons Learned

7.1 Andrew's Thoughts

I learned how much the semantics of a language affects what a programmer is able to do with it. Specifically, we originally had ML as a target language but found the C-style nature of our Setup language made for awkward translation. We shifted to targetting C++ and found the translation significantly more straightforward in most every case. It was not until we started attempting to implement the Set-Builder functionality that our choice of C++ may have hindered our progress. The functional nature of the SetBuilder made C++ less than optimal and we were never able to get it working properly. Additionally, I have become incredibly more familiar with the underlying mechanics of a compiler. And of course I learned OCaml and to start large projects earlier. For others, I would recommend starting early and experimenting with a couple different approaches before deciding on a final plan of action.

7.2 Ian's Thoughts

This project served as my introduction to large group programming projects. If I had to do it all over again, I would have chosen the same group – but I might have approached the project slightly differently. First, I would have insisted on tighter specifications for both the architecture of the compiler and the interface with which the different modules would communicate.

I would have spent much more time with O'Caml. It was only at tthe very end of the project when the sheer power of the pattern matching features became apparent. This occured somewhere right around the time we attempted to "unwrap" arbitrarily nested sets and tuples to generate output (C++) code.

7.3 Bill's Thoughts

Implementing the the type checker was a good exercise in understanding how language feature affect compiler design. Setup supports a compact syntax for generating Cartesian products that we call the Set Builder expressions. While the rest of the language is statically scoped and the compiler can verify type assignments by looking in a single frame of function context, Set Builder has a lambda style syntax in which variables are defined by the result of an expression evaluation. Further more, Set Builder expressions may nest, and variable bindings can occur anywhere along the chain of Set Builder contexts, as well as from the calling function's context.

Consider this example:

```
set A = {1,2,3};
set S = { (x, y) | x in A, y in { z | z in {"a", "b", "c"} } }
```

S will be composed of (int * string) tuples. X will take on an int type, y will take on the type of z, and z will have type string. A is resolved by first search the set builder context, and failing to find it there, looking in the function context. All the other variables are defined in the chain of set builder context.

It should also be noted that our tuple types are deep structures, so that (1, 2) is not of the same type as (1,(2)).

Completing the type checker was a bit tricky, as both selection of the correct frame to make a binding decision and getting the correct level in the type structure was also an issue that was solved.

7.4 Adam's Thoughts

This project was hard, but I enjoyed it a lot.

Chapter 8

Appendix

8.1 Project Log (GitHub)

```
commit 61f2d421751f9545581f6650e3a35a33f64d7445
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 22:36:17 2011 -0500

    commiting edits to the reference manual

commit c46c1f830b039139c79d00c0838476b57e2c06bb
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 22:03:50 2011 -0500

    updated refMan and added .out files for some tests

commit 2861da6cbd5a844da0a7696f8bd1f1fbc83fd1d4
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 21:43:30 2011 -0500

    validation repair

commit 8d8ea993fef8a0fb2513cd713b4ed87ccde018b9
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 21:12:05 2011 -0500

    Fixed Tuples, SetFactory, TupleFactory, SetRange, and printing bools

commit f6a324420dc6be8f88aa10f82ee96cab492dd658
Author: Ian Erb <ianrerb@gmail.com>
```

```
Date:   Thu Dec 22 18:55:45 2011 -0500

    modified lessons learned

commit 4c86259bf837667f5e28399f723400e6b79304eb
Merge: 80a9672 3d648dd
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 18:04:32 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 80a96728c23544400de58d57ef574ccc6804b9fd
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 18:03:33 2011 -0500

    updated RefMan and Removed Bill Miller from slides

commit 3d648dddaa6a8b3ccefb80b82e5fe60bb64809f4
Author: acingraham <acingraham@gmail.com>
Date:   Thu Dec 22 17:31:38 2011 -0500

    Some simple programs I'd like to get working before we submit.

commit d862d2a9b3c524dbe7980a234933e0c792bb895a
Merge: b129ff8 d4d5429
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 11:05:29 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit b129ff83706155624e9bfbeaaa7c520ba5e8805b
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 11:05:11 2011 -0500

    pres

commit d4d5429928c44a814da9460329a09052cdccf3fd
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 10:41:08 2011 -0500

    Added setDiff to demo

commit f63346dbab40b1b486d24254cdadf6b171f04f46
```

```
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 10:21:03 2011 -0500

    Added to demo

commit 3bbd20b1a4fd1ede9e8a3e03918823ba7a846c30
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 09:35:15 2011 -0500

    Altered demo

commit 443346b525c5c3f5a0fecbdf97c13acc98daddef
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 09:18:25 2011 -0500

    setOp to demo

commit de01c5308ae8855808373ef718e54d4562fe1ea8
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 09:16:01 2011 -0500

    Added cardinality

commit 5dd09593223fcc6037be810edfe0d744b2cdba4e
Merge: 17861bf 8c09684
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 22 09:11:29 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 17861bf0c102159fa617858e499476dfc8ab01f6
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 22 09:11:13 2011 -0500

    fix string issue

commit 8c09684fb6bf65b3c0439a5667b328b3d375b1e0
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 08:26:58 2011 -0500

    Demo Folder

commit 87c88c313e9e49c9b9f42b698a44a4d17fe24623
```

```
Merge: a5d2770 34169a6
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 08:20:17 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit a5d2770076ffec25212d1279addeccf419082791
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 08:19:40 2011 -0500

    added su.out files

commit 4de777c402498d0954ab005650767b182b2def1a
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 07:59:10 2011 -0500

    updated docs

commit 34169a6f6d912e2f9131a00127f70ccc446a120c
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 07:30:47 2011 -0500

    Fiddled with setupBase.h

commit 12269b1d9b78d3589600d4b7b98d97f253fbfa2a
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 22 07:28:36 2011 -0500

    match warning

commit c922521d057a94e7cca7b4af53cbc2e828a5708e
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 22 07:24:36 2011 -0500

    sharing some code

commit 521e6c2bd7f4ba1869c0ab05bf90b3a0a8f74a21
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 04:04:41 2011 -0500

    Overloaded != operator for SetupBase types and added another test

commit 8e7e5e2483b963c897088eebec188d11ed6825e0
```

```
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 03:40:13 2011 -0500

    Fixed Intersect so it no longer calls Union

commit 5663cd1f5b10dbbb44c8f3f4f527a99d633fe43d
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Thu Dec 22 03:28:39 2011 -0500

    Changed some tests

commit 93bd643a5bc5b8a7d0cc924c8eded17be09ab344
Merge: d38c15a a563e38
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 22 03:06:47 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit d38c15ac793b639a5ccff36d5e9e9a985a04479a
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 22 03:06:16 2011 -0500

    cross and union fixes

commit a563e38ce6c15209fa5d378f739edc28bfaf5ec5
Merge: 0a81278 d199218
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 03:05:57 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 0a81278f1d11933700271e1c37c67c6cd00c3a7e
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Dec 22 03:02:27 2011 -0500

    updated LRM for new grammar rules, tutorial, etc

commit d199218c3476c12090d863fa508957ddb452fff7
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 23:47:46 2011 -0500

    take care of ml compiler warnings
```

```
commit 93bc812c6e04ac263717a4372a8a31b9b80dd35d
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 23:31:26 2011 -0500

    a test with deeper nesting

commit a06d59f63a7d06e5ff025e2c06dfbb73fcb22856
Merge: 0e1d0f8 bb233a2
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 22:58:38 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 0e1d0f8694e354f8db1e60d4a86b43b6e5167681
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 22:58:08 2011 -0500

    set op bug

commit bb233a2f039eeb1767801bab064275c008f20359
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Wed Dec 21 22:53:22 2011 -0500

    Added setIntersect

commit e753f05624ae9b551c92bb242fe82e5d48ca8fd6
Merge: 2de1936 a15e28c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 22:44:39 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 2de19361a5433a892644b37e4a5049da11157b54
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 22:44:18 2011 -0500

    tests

commit a15e28cf1e944c07bfb01db4344d76af26243751
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Wed Dec 21 22:35:03 2011 -0500

    Corrected set function calls like setUnion, setCross, etc.
```

```
commit 17a8ed7597f0576ed080e1f486e0d7b7d849e74a
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Wed Dec 21 22:26:40 2011 -0500

    Notes on memory leaks

commit d8d89fab8578710d0682741e4db9491174f60222
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Wed Dec 21 22:14:13 2011 -0500

    SetupBool prints "yup" and "nope"

commit 9e1a438d5988004a14155808bc319e1c41b4432f
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 22:06:11 2011 -0500

    set ranges

commit 57a0b034c5d98b0be51d2388932b2f7a11a0bc0b
Merge: 977bce1 b7f2c63
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 21:39:58 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 977bce19071635257352b80b201896afecf71962
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 21:39:43 2011 -0500

    nesting

commit b7f2c631c6b21169f700409a8cda29d19c732f47
Author: unknown <Andrew@Andrew-PC.(none)>
Date:   Wed Dec 21 21:13:02 2011 -0500

    Untested destructors for SetupSet and SetupTuple

commit 8210ff57df48722d3eef7a71cf517188000863f0
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 18:55:11 2011 -0500

    cWriter tuples
```

```
commit 0fa245f8135ede0ec7fda7ae1dcdfa01383dc3c8
Merge: 55e5dd3 ad5f090
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 18:37:26 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 55e5dd358258b727197c132cf3253411a10314af
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 18:36:59 2011 -0500

    cWriter bits

commit ad5f090741c17009324b85ef12cdae91ae549065
Author: acingraham <acingraham@gmail.com>
Date:   Wed Dec 21 18:13:10 2011 -0500

    Added cross

commit 5d7550a2f34cb964f996da4bb0e28c1767d8e8d6
Author: acingraham <acingraham@gmail.com>
Date:   Wed Dec 21 17:52:11 2011 -0500

    Added setUnion, setMinus, setCardinality, and overloaded = operator

commit 17fd4294bc2e2aed3a854f3698f173a6183b1bb5
Merge: b6e7d32 e3c71ed
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 16:40:10 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit b6e7d321ba1218561555d59fcac96fef56ff4c5d
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 16:39:00 2011 -0500

    more tests and fixed NDIVIDE BUG

commit 4e4d0b625d7376243785569ab5f7b8f6067a3760
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 16:33:59 2011 -0500
```

```
    fixed Ndivide bug in cWriter.ml

commit e3c71ed82b29a645a758b407122cce78a947f546
Merge: 5706922 a947eac
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 16:09:48 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 570692264254eec57826eb141f62ce15d2d38c86
Merge: a866522 325025d
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 16:09:11 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

    Conflicts:
     tests/cWriter/expect_success/arithmetic_float_literals.su.err
     tests/cWriter/expect_success/arithmetic_float_vars.su.err
     tests/cWriter/expect_success/arithmetic_int_literals.su.err
     tests/cWriter/expect_success/arithmetic_int_vars.su.err

commit a947eacdd45529df6bb3915d6ab34135f6c8ca64
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 15:58:52 2011 -0500

    arithmetic and comment tests. *NDIVIDE BUG FOUND* need to cast whole expression to i

commit a866522db2797cbebab0491ccac69f7d63601c9a
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 15:55:19 2011 -0500

    functions can return sets

commit 325025de68956f90ca9cff80d00bb0ccb3a727b9
Merge: be5c5fb 604bb12
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 14:57:15 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 604bb12e4842a85277224f1e3b79d6ee4dc53367
Author: Bill Warner <whw2108@columbia.edu>
```

```
Date:   Wed Dec 21 14:54:13 2011 -0500

    some cWriter tests

commit be5c5fbe5158e1290359299ba13536f78a8508bd
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 14:51:29 2011 -0500

    literal and float arithmetic tests

commit 9a067b4b259d1b0e02ad50b32ee2633b8d2e42cf
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 14:07:11 2011 -0500

    fixed tests for linux

commit e236c396ef319fdb9df9d54ca2a2f212320fba71
Merge: 9e2eabc b7d753a
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 13:47:37 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 9e2eabc04a1e04894d0c815b7e73031195c54e5d
Merge: bb6b001 9c39590
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 13:44:26 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit b7d753afd1e4fa9ccdbbd7c8979b7bdb17fec8ee
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 13:43:36 2011 -0500

    cWriter tests folded into Makefile

commit 9c39590ae66ea6a29e3b5eb8bc551c5197a3fca9
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 13:08:32 2011 -0500

    print statement, fixed add test for success

commit bb6b0015018fb8dca87514a0f370ab07aa4ab3c6
```

```
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 12:51:31 2011 -0500

    updates to refMan

commit d9e458de6a2c8266edb42c8b5c5a759bbea54dc0
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 12:30:10 2011 -0500

    deep comparison for set assignment

commit 0492e9b846950608c0d1765941ee45ccbf5ebe21
Merge: 93e43f1 6e57d9f
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 12:15:56 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

    Conflicts:
     validation.ml

commit 93e43f17cb45059c24ace81e6ac08f18fd7ce1df
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 21 12:11:12 2011 -0500

    better setbuilder type checking

commit 6e57d9f9710ba209d0e91dd4fb5c50227e34e80a
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 11:25:02 2011 -0500

    added auxiliary files for refMan to .gitignore

commit 8ee1bef4d340efa2244e4b7612e6877af2f0099a
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 11:22:45 2011 -0500

    edits to refMan, adding back validation.ml file

commit febdf69683e01b02f994f771e850d7454b0dd777
Merge: 931868c 721336b
Author: Ian Erb <ianrerb@gmail.com>
Date:   Wed Dec 21 10:23:11 2011 -0500
```

```
    Merge branch 'master' of github.com:weissadam/plt-fall-2011

    Conflicts:
     validation.ml

commit 721336b642258fc293579295e119adab3d67d67f
Merge: 0300111 8da9346
Author: Bill Warner <whw2108@columbia.edu>
Date:   Tue Dec 20 22:39:07 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 03001112e7c523a2117dacbca6744eea33bd01df
Author: Bill Warner <whw2108@columbia.edu>
Date:   Tue Dec 20 22:38:19 2011 -0500

    set expression semantics (whew!)

commit 8da9346bbe4c340d88eb332fdf37ffd4c42a6bd9
Author: acingraham <acingraham@gmail.com>
Date:   Tue Dec 20 18:39:42 2011 -0500

    Added clone and copy constructor

commit f41b03a583f725ff5efc8977396f6db2e440a962
Author: acingraham <acingraham@gmail.com>
Date:   Tue Dec 20 17:03:36 2011 -0500

    Added a tests folder and some tests for setupBase.h.  I haven't made any changes to

commit dadf63e626749cf560500f6c083128ca9cb667f2
Author: acingraham <acingraham@gmail.com>
Date:   Tue Dec 20 16:53:33 2011 -0500

    Fixed not allowing duplicate entries in Sets.  I left it out when I integrated with

commit 8518ead580ea1e545bf06a1a725fdae03e5ddf43
Author: acingraham <acingraham@gmail.com>
Date:   Tue Dec 20 16:18:52 2011 -0500

    Overloaded == operator for SetupBase. Added isDuplicate function to SetupSet and Set
```

```
commit 1702f440a52e063a261d2d6f80e3603a898452c4
Merge: a61d005 b747e07
Author: Bill Warner <whw2108@columbia.edu>
Date:   Tue Dec 20 11:04:08 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

    Conflicts:
     setupBase.h

commit a61d00589676d6a74eb97cec849ec724dd11a35c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Tue Dec 20 10:54:12 2011 -0500

    chaining add

commit b747e07e5acb48ac1253fcd1faeaf25cfa455b60
Author: acingraham <acingraham@gmail.com>
Date:   Tue Dec 20 10:46:56 2011 -0500

    Overload << operator for SetupBase.h

commit 3b812d36b0f2970b9b57c08e6c55463324aa208c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Tue Dec 20 07:09:44 2011 -0500

    quote strings in setupBase

commit 931868cb06515c0b08151285afbac21e84bdfd04
Author: Ian Erb <ianrerb@gmail.com>
Date:   Mon Dec 19 23:48:41 2011 -0500

    cWriter and Print stmt and tests

commit e34b19625c39d788c8bc368e9f95d7d5c61f305c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Mon Dec 19 23:36:30 2011 -0500

    cpp setup types

commit 5b6d2e8731cffb28ff2586f0c50f302fdb75cb85
Author: acingraham <acingraham@gmail.com>
Date:   Mon Dec 19 17:43:10 2011 -0500
```

```
    Fixed assignment and declaration so chaining is possible.

commit dd0a0aa78753fa50661729d0dd42b10b34726af3
Author: Bill Warner <whw2108@columbia.edu>
Date:   Mon Dec 19 15:55:54 2011 -0500

    code gen tests

commit c6208edec4ed00f16a54abd675f51c87d5d3d768
Author: Bill Warner <whw2108@columbia.edu>
Date:   Mon Dec 19 15:00:12 2011 -0500

    preserve order of function args

commit 6cd0b2b29c495428deed738d6e18813d64cfed39
Author: Bill Warner <whw2108@columbia.edu>
Date:   Mon Dec 19 12:20:20 2011 -0500

    while statement

commit ecb04778b73843decd6594b1b0a5f31bfab0734f
Merge: d287d78 0eefa18
Author: Bill Warner <whw2108@columbia.edu>
Date:   Mon Dec 19 08:51:42 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 0eefa1818cf076eacbdecfec449fa039269e5ac0
Author: acingraham <acingraham@gmail.com>
Date:   Mon Dec 19 05:43:58 2011 -0500

    Added while, assignment, set literal, set range, mutable variables, variable default

commit d287d78629acf290939677fa14b8d3606dfbdd62
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 21:05:42 2011 -0500

    added type checking line to code generator

commit 647583703a8e62b3aeffea7270a0a666adbbd91b
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 20:33:42 2011 -0500
```

```
    codeGen tests added to run_tests.py

commit a408b0a2dbb28856a318cd4bf00cd1d0c9091c42
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 20:14:20 2011 -0500

    If statments and fixed a bug

commit c38a4ca6b7bd4b7c113bfdf257b8caea2afae7a5
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 18:40:54 2011 -0500

    changed the word 'scoping' to 'validation'

commit aff193e06a9ae00e318b51c9eab42ae3d9793ddb
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 17:59:39 2011 -0500

    guarantees a return statement

commit 1e8dddee0a590dcf0fcb5b3a745d85e569092951
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 16:25:15 2011 -0500

    some type checking for set builder

commit 313910578910bf1051bd0f43618bf85d38885743
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 14:04:21 2011 -0500

    set range

commit 5433071ba3071cb76bd052920b7c48d91f3ec622
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 13:41:28 2011 -0500

    unary ops, removed warnings

commit d3942416bd23d40583aa333158f3335bf03638eb
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 13:04:11 2011 -0500
```

```
    fixed a bug hidden by test system

commit 74b40fdf77503143bc2f79ae9c5b770466db320d
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 12:34:57 2011 -0500

    better testing; each test is composed of three files: test.su test.su.out and test.s
    When you have created a test that is passing, save it using this example:

     $test_exec < $F > $F.out 2> $F.err

commit 4abe70226b3a27eed0072507958af46dabde3c3c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 01:59:47 2011 -0500

    fixed bug in tuple parsing

commit d1e194bd05747216ca0b64519d9b2ad586d288cf
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 00:59:53 2011 -0500

    these tests no longer apply

commit f66f8b77d0e21087e29607dcb1a3c2431661f0ee
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 18 00:55:55 2011 -0500

    tuples, actually checked in.

commit fb5a05f85180aaf65a8cfee51c049d5c69dd2fe8
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 17 23:47:09 2011 -0500

    tuples

commit 807ffc59d4946f82c3c136ec8a5811147a23338f
Author: Adam Weiss <adam@signal11.com>
Date:   Sat Dec 17 22:19:47 2011 -0500

    Parser: Global variables decl outside of funcs

    NOTE:  Ast.program is now a record type.
```

```
    Given an instance of Ast.program p:

    p.funcs is the list of funcdefs that Ast.program once was
    p.globals is a list of statements that are guaranteed to
    only be of the type Decl() with NullExpr for the expression.

commit a15e273fb601751f8f7343134eddc23602d7063e
Author: acingraham <acingraham@gmail.com>
Date:   Sat Dec 17 19:25:20 2011 -0500

    Added funcalls, if-else, boolean operators, and more test files to codegen.

commit 70ef5db8d4bc50704149978ea16b8b67d8c25061
Merge: 8b0218d 221cd03
Author: acingraham <acingraham@gmail.com>
Date:   Sat Dec 17 19:20:46 2011 -0500

    Merge branch 'master' of https://github.com/weissadam/plt-fall-2011

commit 8b0218d3bf2ada264add552bc1fae516653fa860
Author: acingraham <acingraham@gmail.com>
Date:   Sat Dec 17 19:18:16 2011 -0500

    Added funcalls, if-else, boolean operators, and two more tests to codegen

commit 221cd038b9a1810599d9268323cc86faf7e40099
Merge: 637a2e2 7477649
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 17 18:45:15 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 637a2e2f5354d401821ed525c98356feed8b316e
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 17 18:43:01 2011 -0500

    function return type checking

commit 7477649b3576e564164154ec3cb3253acdb483df
Author: Ian Erb <ianrerb@gmail.com>
Date:   Sat Dec 17 16:36:14 2011 -0500

    code Generator and output test directory
```

```
commit b5ee84f41e9b2a43fadb70f5ab6ca0c8df327288
Merge: 02a223a 7f9dd05
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 17 08:48:08 2011 -0500

    Merge branch 'master' of github.com:weissadam/plt-fall-2011

commit 02a223aae3c29fcedfb44b0ed8a4011b74607cd1
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 17 08:47:37 2011 -0500

    conform to spec

commit 7f9dd0573001aeefb1bfc61436e5914894009c6d
Author: Ian Erb <ianrerb@gmail.com>
Date:   Fri Dec 16 18:56:47 2011 -0500

    updated refMan for ranges

commit 9830e7634b433050ce54841f7d336fb91c944114
Author: Ian Erb <ianrerb@gmail.com>
Date:   Fri Dec 16 18:47:59 2011 -0500

    updated refMan for scoping and global vars

commit b298eed84103e0d49c7193692e4dd972995c9f53
Author: Ian Erb <ianrerb@gmail.com>
Date:   Fri Dec 16 14:22:22 2011 -0500

    added presentation shell and first slide

commit 7181293794f06edd5ea7bbe9c388c9047f98c3ed
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 13:00:22 2011 -0500

    coupla more interesting tests

commit 898d4ed99deccdd7881fedbbf8a84ccf9ae0c391
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 12:49:13 2011 -0500

    comments
```

```
commit 617a57b44fb2dcc35d3eba7e7e9440f29f85a07d
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 12:26:03 2011 -0500

    corresponding success test

commit 2d0dbf6eeca46130b40ce02632d2da32ee5e4cde
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 12:20:00 2011 -0500

    taller stack still passes

commit 50b084db4b07ed221b9cfeb596c2ea011111bc2d
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 12:14:34 2011 -0500

    fixed stack traversal

commit b1d79bf81063517aecab5da59bc2c0a1067a6b95
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 11:56:38 2011 -0500

    important test fails

commit e72362e56818969407b80d072b1aa94c85c13d9c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 11:41:43 2011 -0500

    forgot to push the frame

commit 738ba27f8f5ffe4b9ffe2b5172e6346f347473cc
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 11:28:21 2011 -0500

    var scoping in function calls!

commit ac354c4093e534e849bd4c279dc6745010477633
Author: Bill Warner <whw2108@columbia.edu>
Date:   Thu Dec 15 10:43:58 2011 -0500

    better naming for function context
```

```
commit 45e42af35abf5ba59ffec9195bf5ee8eead83944
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 14 22:13:11 2011 -0500

    less redundant

commit 6977334d373cdf47e0bbfd130c7dc8e03842a0ca
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 14 21:36:55 2011 -0500

    defined a symbol table type; part way into checking types of variables in functions

commit 312eead4e978c8444fe7f000be96be11723f0eee
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 14 10:26:30 2011 -0500

    bit of progress with function call scoping

commit d13ed60ad540e6b30286407636ed2605fb1eebfe
Author: Bill Warner <whw2108@columbia.edu>
Date:   Tue Dec 13 10:56:33 2011 -0500

    test for set literal

commit 229c67f555adf88b176fd6ae721afb8ecbdb0f95
Author: Bill Warner <whw2108@columbia.edu>
Date:   Tue Dec 13 10:54:26 2011 -0500

    homogeneous typed sets

commit 9199b7ad7517bdddca508ae16e786e10f31c822c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Tue Dec 13 10:28:09 2011 -0500

    removed redundant fdef for minus, re-introduced NoType

commit 029ad58313b5d3ca5ac02d3d40514a15e666de28
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sun Dec 11 15:40:29 2011 -0500

    set literal test (fails)

commit fc4b770e4260a36939db7f8be864a79e57b254b9
```

```
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 10 22:57:43 2011 -0500

    assignment type checking

commit d740d7214ab348a785266259d72b7af38e31b35e
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 10 21:03:20 2011 -0500

    some tests

commit b8302e00a0a9798e00f5c9ab048710812a35eb0c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 10 20:33:32 2011 -0500

    test output

commit a920e0b6280db90318eb8d41547b18770db29b2d
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 10 20:31:49 2011 -0500

    a few tests

commit 0f1f7835b3cbd1bc4d6bb26523ade093a3751b33
Author: Adam Weiss <adam@signal11.com>
Date:   Sat Dec 10 20:07:53 2011 -0500

    Type checking!

commit c00df82cd1af439b4d282b6ecd17be8fce69bbf9
Author: Adam Weiss <adam@signal11.com>
Date:   Sat Dec 10 16:31:16 2011 -0500

    Fix parser problem with declaration/assignment

commit 1901ca4a253b82e182dda8fedf7ca1c914105672
Author: Adam Weiss <adam@signal11.com>
Date:   Sat Dec 10 16:20:10 2011 -0500

    Support for "make test" and fix path in run_tests

commit d73071f6b3dc27d3d8c02c9690da4b7eb7153861
Author: Adam Weiss <adam@signal11.com>
```

```
Date:   Sat Dec 10 16:19:11 2011 -0500

    Add gitignore

commit e1d4293d40822dae27aad1557f380e4552adf7a8
Author: Adam Weiss <adam@signal11.com>
Date:   Sat Dec 10 16:18:40 2011 -0500

    Support C style comments and fix test

commit 0ce870edf36898f93e1343310b649f627dcac4e3
Author: Bill Warner <whw2108@columbia.edu>
Date:   Sat Dec 10 10:00:18 2011 -0500

    some type inference code, uncalled

commit ae8b3fe515930d1cba506d53cd3e1f136f1b960a
Author: Bill Warner <whw2108@columbia.edu>
Date:   Fri Dec 9 19:39:17 2011 -0500

    get the tests in there

commit 37031a4fc6dc26f4d1fd06511e8fbbf127107d46
Author: Bill Warner <whw2108@columbia.edu>
Date:   Fri Dec 9 12:19:52 2011 -0500

    comments for run_tests.py

commit a03174c7536e6f26b4e8970b894e9ea57ed662ad
Author: Bill Warner <whw2108@columbia.edu>
Date:   Fri Dec 9 12:07:02 2011 -0500

    replaced passed/failed with yup/nope

commit f2afe2872a40d46fb404c602bc4d7878fa256522
Author: Bill Warner <whw2108@columbia.edu>
Date:   Fri Dec 9 12:03:12 2011 -0500

    discover test types

commit a9258faaaeeea2d92e1237f446bffbbc96c823bb
Author: Bill Warner <whw2108@columbia.edu>
Date:   Fri Dec 9 11:58:47 2011 -0500
```

```
    refactored includes, test harness

commit 0a1bf7df050ca476e2da68b295a359a3511fc271
Author: Bill Warner <whw2108@columbia.edu>
Date:   Fri Dec 9 11:04:36 2011 -0500

    raise exception for multiple declarations

commit 0b21a8e570fdfc13b4d68e7ed5125cae597e38ef
Author: Bill Warner <whw2108@columbia.edu>
Date:   Fri Dec 9 10:46:52 2011 -0500

    We infer SetType for assignments in expressions

commit 727cb6907c242e103a4791a2dfe3f8165e6c1ca5
Author: Bill Warner <whw2108@columbia.edu>
Date:   Fri Dec 9 03:03:52 2011 -0500

    some var scoping

commit 6f7dcfff8be6fcc6c233e86cfdf66129ee7f813c
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 7 23:13:24 2011 -0500

    messaging

commit 72e8e2c85e03aa67eafc8c369c3f85f8acda2ff3
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 7 22:23:29 2011 -0500

    rm'd some dumb files

commit e217ed46713c17ed743cc0f182abfa5d25bef762
Author: Bill Warner <whw2108@columbia.edu>
Date:   Wed Dec 7 22:20:57 2011 -0500

    some tests and a test harness in python

commit c3e8771e26fb60576f63dbf0788a577098bcd1a5
Author: Adam Weiss <adam@signal11.com>
Date:   Mon Nov 21 19:28:15 2011 -0500
```

```
    First pass at a parser

    This is really not clean enough for a commit, but it's at least
    something to share with the team.

commit 45636477bc52908082f0a0102596ea393c53e215
Author: Adam Weiss <adam@signal11.com>
Date:   Mon Nov 21 19:24:31 2011 -0500

    Move doc off before first commit

commit 5556065ceeea588f07cda18820f794343e4196e5
Author: Ian Erb <ianrerb@gmail.com>
Date:   Mon Oct 31 21:13:37 2011 -0400

    added author names to front page

commit fa415ad2da67e10e835653a54bb49fb49dea0ae7
Author: Ian Erb <ianrerb@gmail.com>
Date:   Sun Oct 30 21:01:16 2011 -0400

    complete first draft for submission

commit 5fa5fb4dcc641ae0ffe275700553fbacab857eac
Author: Bill Warner <bill@dyn-209-2-218-7.dyn.columbia.edu>
Date:   Fri Oct 28 00:12:49 2011 -0400

    remove test file

commit 75649879a845bdd0b30fb6e442bd5a4ab1fc5583
Author: Bill Warner <bill@dyn-209-2-218-7.dyn.columbia.edu>
Date:   Fri Oct 28 00:12:24 2011 -0400

    test push

commit e9c7dd7e3609fd2ebc8c842173ffbf18042eb720
Author: Ian Erb <ianrerb@gmail.com>
Date:   Thu Oct 27 23:56:51 2011 -0400

    variables, grammar, other

commit 0c1f1690de3c7dfdb4fb3044cdda7c48b5b51559
Author: Adam Weiss <adam@signal11.com>
```

```
Date:   Thu Oct 27 22:17:45 2011 -0400

    create repo
```

8.2 File Listing

8.2.1 ast.mli

```
type typespec =
    IntType
  | FloatType
  | StringType
  | TupleType
  | SetType
  | BoolType
  | NoType
  | TupleSeq of typespec list
  | SetSeq of typespec


type binop =
    Plus
  | Minus
  | Times
  | Divide
  | NDivide
  | GrThan
  | LeThan
  | GrThanEq
  | LeThanEq
  | Equals
  | NotEquals
  | LogicalAnd
  | LogicalOr
  | Union
  | Cross
  | SetMinus
  | Intersect

type unop =
    Negative
  | Not
  | Cardinality

type expr =
    NullExpr
  | Assign   of string * expr
  | Binop    of expr * binop * expr
  | Unop     of unop * expr
  | Int      of int
  | Boolean  of bool
  | Float    of string
  | String   of string
```

8.2.2 scanner.mll

```
{ open Parser }

(* for floats *)
let digit = ['0'-'9']
let sign = ['+' '-']
let exp = ('e' sign? digit+)

rule token = parse
(* eat whitespace *)
  [' ' '\t' '\r' '\n'] { token lexbuf }

(* eat comments *)
| "/*"        { comment lexbuf }

(* funcdefs *)
| "function"  { FUNCTION }
| '{'         { LBRACE }
| '}'         { RBRACE }
| '['         { LBRACKET }
| ']'         { RBRACKET }
| "returns"   { RETURNS }
| "return"    { RETURN }

(* types *)
| "int"       { INT }
| "set"       { SET }
| "float"     { FLOAT }
| "string"    { STRING }
| "tuple"     { TUPLE }
| "bool"      { BOOL }

(* binops *)
| '+'         { PLUS }
| '-'         { MINUS }
| '*'         { TIMES }
| '/'         { DIVIDE }
| "//"        { NDIVIDE }
| '<'         { LETHAN }
| '>'         { GRTHAN }
| "<="        { LETHANEQ }
| ">="        { GRTHANEQ }
| "=="        { EQUALS }
| "!="        { NOTEQUALS }
| "&&"        { LOGICALAND }
| "||"        { LOGICALOR }
| "union"     { UNION }
| "cross"     { CROSS }
```

8.2.3 validation.ml

```
open Ast
open Stringofsetups
open Setuptypes


let empty_sc =
  let fsig = {fformals = []; freturn = NoType} in
  let fvars = (Hashtbl.create 3) in
  FunctionContext({ fname = ""; fsig = fsig;  fvars = fvars; has_return = false})

let rec var_collide var_name symbols =
  match symbols with
  | [] -> false
  | hd :: tl -> if Hashtbl.mem hd var_name then true else var_collide var_name tl

let var_add var_name var_type ifcl =
  match ifcl with
  | FunctionContext(fc) -> Hashtbl.add fc.fvars var_name var_type
  | Callers(fc, fcl) -> Hashtbl.add fc.fvars var_name var_type

exception UndeclaredVariable of string

let rec var_get var_name fcl =
  match fcl with
  | FunctionContext(fc) ->
       (try Hashtbl.find fc.fvars var_name
       with _ -> raise(UndeclaredVariable(var_name)))
  | Callers(c, cl) ->
        try Hashtbl.find c.fvars var_name
        with _ -> var_get var_name cl

exception MultipleDecls of string * string

let validate_formals formals fcl =
  let ctx =
    match fcl with
    | FunctionContext(c) -> c
    | Callers(c, _) -> c
  in
      List.iter (fun pair -> Hashtbl.add ctx.fvars (snd pair) (fst pair)) formals;

exception Unimplemented of string
exception TypeError of string
exception NotASet of string

let compute_return_type_plus lhs rhs =
    match lhs, rhs with
```

8.2.4 parser.mly

```
%{ open Ast %}

%token EOF ASSIGN SEMI PIPE IN ELLIPSIS YUP NOPE

/* funcdefs */
%token FUNCTION LBRACE RBRACE LBRACKET RBRACKET RETURNS RETURN

/* types */
%token INT SET FLOAT STRING TUPLE BOOL

/* binops */
%token PLUS MINUS TIMES DIVIDE NDIVIDE LETHAN GRTHAN LETHANEQ GRTHANEQ
%token EQUALS NOTEQUALS LOGICALAND LOGICALOR UNION CROSS SETMINUS INTERSECT

/* unops */
%token POUND NOT

/* statements / funcalls */
%token LPAREN RPAREN COMMA WHILE IF THEN ELSE PRINT

%token <int>    INT_LITERAL
%token <string> STRING_LITERAL
%token <string> FLOAT_LITERAL
%token <string> ID

%right ASSIGN
%left PLUS MINUS
%left EQUALS NOTEQUALS LOGICALAND LOGICALOR UNION CROSS SETMINUS INTERSECT
%left TIMES DIVIDE NDIVIDE
%left NOT POUND
%left LETHAN GRTHAN LETHANEQ GRTHANEQ
%left UMINUS

%start program
%type < Ast.program> program

%%

program:
  /* empty */ { { funcs = []; globals = [] } }
  | func_def program { { funcs = $1::$2.funcs; globals = $2.globals } }
  | var_decl program { { funcs = $2.funcs;     globals = $1::$2.globals } }
```



```
var_decl:
  typespec ID SEMI  { Decl($1, $2, NullExpr) }

stmt_list:
```

8.2.5 cWriter.ml

```
open Ast
open Validation
open Setuptypes

open Printf

(* Authors: Andrew Ingraham, Bill Warner, Adam Weiss *)

exception Unimplemented of string
exception BindingError of string

(* Prevents collisions with library functions *)
let funcname_mangle f = match f with
 | "main" -> f
 | _ -> "_" ^ f


let string_of_typespec = function
   IntType     -> "int"
 | SetType     -> "SetupSet"
 | FloatType   -> "float"
 | StringType  -> "std::string"
 | TupleType   -> "tuple"
 | BoolType    -> "bool"
 | NoType      -> "notype"
 | TupleSeq(_) -> "SetupTuple"
 | SetSeq(_) -> "SetupSet"

let string_of_binop = function
   Plus  -> "+"
 | Minus -> "-"
 | Times -> "*"
 | Divide -> "/"
 | NDivide -> "//"
 | GrThan  -> ">"
 | LeThan  -> "<"
 | GrThanEq -> ">="
 | LeThanEq -> "<="
 | Equals   -> "=="
 | NotEquals -> "!="
 | LogicalAnd -> "&&"
 | LogicalOr  -> "||"
 | Union -> "union"
 | Cross -> "cross"
 | SetMinus -> "minus"
 | Intersect -> "intersect"
```

8.2.6 setup.ml

```
open Ast
open Stringofsetups
(* Author Adam Weiss *)
let _ =
  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
  let result = string_of_program program in
  print_endline (result)
```

8.2.7 setupBase.h

```
#include <set>
#include <list>
#include <string>
#include <sstream>
#include <iostream>
#include <typeinfo>

/*
  Authors: Andrew Ingraham, Bill Warner */

class SetupBase {
 public:
  //  virtual std::ostream toStream(std::ostream) const;
  virtual std::string toString() const = 0;
  virtual bool operator==(const SetupBase &) const = 0;
  virtual bool operator!=(const SetupBase &) const = 0;
  //virtual SetupBool& operator==(const SetupBase &) const = 0;
  //virtual SetupBool& operator!=(const SetupBase &) const = 0;

  virtual SetupBase* clone() const = 0;
};

class SetupBool : public SetupBase {
 public:
  bool value;
 SetupBool(bool v) : value(v) {}
 SetupBool() : value(0) {}
 SetupBool(const SetupBool& r) : value(r.value) {}
  std::string toString() const {
    std::stringstream out;
    if(value){
            out << "true";
    }else{
            out << "false";
    }
    return out.str();
  }

  SetupBase* clone() const {
        return new SetupBool(*this);
  }

  friend std::ostream& operator<< (std::ostream &out, SetupBool &rhs) {
        out << rhs.toString();
        return out;
  }
```

8.2.8 setuptypes.ml

```
open Ast
open Stringofsetups
(* Author Bill Warner *)
exception ContextError of string

type function_signature = { fformals : (typespec * string) list; freturn : typespec }

type function_context = { fname: string; fsig : function_signature; fvars : (string, typ

type function_context_link = FunctionContext of function_context | Callers of function_c

type execution_context = { evars : (string, string) Hashtbl.t; }

type execution_context_link = ExecutionContext of execution_context | ECallers of execut

let push_fc fc fcl =
  match fc with
  | FunctionContext(c) -> Callers(c, fcl)
  | Callers(c, l) -> raise(ContextError("do you really want to push a call stack onto a

let push_ec ec ecl =
  match ec with
  | ExecutionContext(c) -> ECallers(c, ecl)
  | ECallers(c, l) -> raise(ContextError("do you really want to push a call stack onto a

type symbol_table = { functions : (string, function_context_link) Hashtbl.t }

let print_fc ?(indent="") fc =
  print_endline (indent ^ "fname: " ^ fc.fname);
  Hashtbl.iter (fun k v -> print_endline(indent ^ k ^ " " ^ string_of_typespec v))
    fc.fvars

let rec print_fcl ?(indent="") fcl =
  match fcl with
  | FunctionContext(fc) -> print_fc fc ~indent:indent
  | Callers(fc, fcl) -> print_fc fc ~indent:indent; print_fcl fcl ~indent:(indent ^ "  "

let rec print_typespec ?(indent="") tup =
  match tup with
  | TupleSeq(l) ->
      print_endline("(");
      let ind = indent ^ "  " in
      List.iter (fun t -> print_typespec ~indent:ind t) l;
      print_endline ")"
  | _ -> print_endline( indent ^ (string_of_typespec tup))
```

8.2.9 validationtest.ml

```
open Ast
open Stringofsetups
open Validation
open Setuptypes

let _ =
  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
  let symbols = validate_program program in
  Hashtbl.iter (
  fun fname fcl ->
    (* print_fcl fcl *)
    let fctx =
      match fcl with
      | FunctionContext(c) -> c
      | Callers(c, l) -> c in
    Hashtbl.iter
      (fun k v -> print_endline (k ^ " " ^ (string_of_typespec v)))
      fctx.fvars
      )
    symbols.functions
```

8.2.10 MAKEFILE

```
TARGETS=setup validationtest cWriter
CPP_TARGETS=setupBase.o
NATIVE_TARGETS=$(TARGETS:=.native)
DEBUG_TARGETS=$(TARGETS:=.d.byte)
CFLAGS=-annot
OCAMLBUILD=ocamlbuild -cflags $(CFLAGS)
RM=rm
PYTHON=python

all:        $(CPP_TARGETS)
        $(OCAMLBUILD) $(NATIVE_TARGETS) $(DEBUG_TARGETS)

clean:
        $(RM) -rf _build
        $(RM) -f *.native
        $(RM) -f *.d.byte

test:        all
        cd tests && $(PYTHON) run_tests.py

setupBase.o:
        g++ -g -O -Wall -c setupBase.h
```