CLAM: The Concise Linear Algebra Manipulation Language

Jeremy Andrus and Robert Martin and Kevin Sun and Yongxu Zhang
{jca2119, rdm2128, kfs2110, yz2419}@columbia.edu

Columbia University
COMS W4115: Programming Languages and Translators

December 22, 2011

Contents

Chapter 1

Introduction

CLAM is a linear algebra manipulation language specifically targeted for image processing. It provides an efficient way to express complex image manipulation algorithms through compact matrix operations. Traditional image processing is performed using a language such as C, or C++. Algorithms in these languages are quite complex and error-prone due to the large number of lines of code required to implement something as conceptually simple as, "make this image blurry." The complexity arises from the need to perform elaborate calculations on every pixel in an image. For example, to blur an image you first need to calculate the luminance of the pixel (from the red, green, and blue channels), then you need to mathematically combine this with the luminance of adjacent pixels, and finally re-calculate red, green, and blue values for an output image.

CLAM simplifies image processing, and more generally linear algebra, through domain-specific data types and operators. A basic data type in CLAM is a `Calc`, which can either be a *Matrix* or a *CString*. Matrices can be variable sized with an optional rational coefficient, and are used in image convolution operations. CStrings are simple calculations based on previously defined channels and basic C math operations such as `sqrt` or `atan`.

An `Image` is another CLAM data type which is expressed as a collection of channels. For example, when reading an image into memory, CLAM creates a *Red*, *Green*, and *Blue* channel automatically. Additional Image channels can be assigned from other images, or calculated using an expression syntax that defines a calculation involving the values of other, previously defined, channels. The basic image processing operator in CLAM is the convolution operator. This operator takes an `Image` channel and a `Kernel`, an ordered collection of `Calc` objects. This operator convolves each `Matrix` within the `Kernel` with the input channel, and runs each CString calculation on the input channel in the order they were defined in the `Kernel`. It then collects the resulting output channels into a new `Image`.

Two primary use cases of CLAM are basic image information extraction, and filtering. The compact syntax and powerful basic data types of CLAM make information extraction, such as finding all the edges in an image, simple, compact, and easy to read.

Chapter 2

Language Tutorial

2.1 Input and Output

CLAM's basic I/O operators are `imgread` and `imgwrite`, and every program will have to call them at least once each to do anything useful. `imgread` takes a filename (or integer, see below) as its sole argument and returns an `Image`. `imgwrite` take an `Image`, a format, and filename (or integer).

2.1.1 Your first program

Using only I/O operators, you can already write a simple program that copies an image from one location to another. Or, if the output is in a different format than the input, you have a simple image converter. In either case, you only need two lines of code!

```
Image input = imgread("source.jpg");
imgwrite(input, "png", "dest.png");
```

...or 1 if you're tricky

```
imgwrite(imgread("source.jpg"), "png", "dest.png");
```

2.1.2 Using command-line arguments

This program only copies from `"source.jpg"` to `"dest.png"` which is not very useful. You'd need to edit the code and recompile to change the source and destination. To avoid this problem, `imgread` and `imgwrite` can both be called with integers which refer to items in the command-line argument list, giving your code much greater flexibility. (CLAM will automatically enforce the correct number of command-line arguments.)

```
imgread(1); /* Get filename from argv[1]*/
imgwrite(input, "png", 2); /* Get filename from argv[2]*/
```

2.2 Compiling and Running Your Program

In order to generate binaries from your CLAM program, you will need the `g++` compiler installed and available in your default `PATH`. This is because CLAM uses C/C++ as its compile target, and leverages existing C/C++ compilers to generate and optimize machine-dependent code. You can compile your clam program by simply passing your file to `clam` as the sole argument. This will automatically output a binary to `a.out`. You can also specify the name of the binary with the `-o` flag, and pass in the source file path with `-i`. For example:

- `./clam prog1.clam`
- `./a.out source.jpg dest.png`
- `./clam -i prog1.clam -o copyimg`
- `./copyimg source.jpg dest.png`

The full set of options to the CLAM compiler can be found using the *–help* option, and is reproduced below:

```
CLAM v0.1
Usage: clam {options} [{<} inputfile]
Options are:
  -o <filename> Specify the output file
  -i <filename> Specify the input file
  -c            Output generated C only
  -t            Print AST debugging information
  -help         Display this list of options
  --help        Display this list of options
```

2.3 Basic Types

2.3.1 Channels

Channels are arrays of values associated each pixel in an `Image`. For example, each pixel in an `Image` usually has *Red*, *Green*, and *Blue* values associated with it, and we can further define *Luminosity*, *Hue*, *Saturation* etc. Thus we can refer to the *Red* channel or the *Saturation* channel of an `Image`.

When first read into CLAM, `Images` come with three default Channels: `Red`, `Green`, and `Blue`. These can be accessed using the `:` (colon) operator. The values in one Channel can be copied to another using the `=` (equals) operator. If the Channel on the left-hand side is undefined, it is created dynamically.

The following program uses a `temp` channel to swap the `Red` and `Blue` values of an image:

```
Image img1 = imgread(1);

img1:temp = img1:Blue;
img1:Blue = img1:Red;
img1:Red = img1:temp; /* swap channels */

/*Only Red, Green, and Blue channels are written:*/
imgwrite(img1, "jpg" ,2);
```

2.3.2 Calculations

While the equals operator is enough to create new channels that are copies of old ones, `Calc` objects allow you to create new Channels via calculation. The `:=` (colon-equals) operator is used to define `Calc`s object. Once defined, a `Calc` object cannot be redefined. A `Calc` can be assigned an *atomic type* such as `<Uint16>` or `<Angle>`, and all values resulting from that calculation will be clamped to the appropriate range. The default type is `<Uint8>`, which corresponds to a range of 0-255.

`Calc` objects can be defined in two ways - as *escaped-C strings* (*CStrings*) or as *matrices*.

CString `Calc`s are enclosed in `#[...]#` brackets. These strings can contain basic mathematical operators and functions, as well as references to other Channels. A `Calc` defined in this manner can be applied to an `Image` using the `|=` (or-equal) operator, provided that the `Image` has all the requisite Channels, thereby creating a new Channel with the same name as the `Calc`. The values of this new Channel will be calculated according to the contents of the string. (It follows that anonymous `Calc`s are not allowed.)

```
/* Define a calculation for Luminosity */
Calc Lum<Uint8> := #[(3*Red + 6*Green + 1*Blue) / 10]#;
Calc Zero := #[0]#;

Image srcimg = imgread(1);
/* Add luminosity channel to the Image */
srcimg |= Lum;

/* Add a 'black' channel to the Image, named 'Zero' */
srcimg |= Zero;

/* The following is invalid - no name! */
srcimg |= #[Red + Green + Blue]#;

/* Calcs cannot be redefined! */
Lum := #[Red * Green * Blue]#
```

Matrix `Calc`s can be of any size, and are represented as lists of numbers enclosed in `{...}` braces. Rows are separated by commas, and the *Matrix* is optionally preceded by a scaling factor of the form `[` *numerator* `/` *denominator* `]`. Matrices represent a weighted (and scaled) sum of values in the neighborhood of a pixel. Because matrix `Calc`s cannot be calculated with a simple "for loop," over all pixels, they cannot be applied directly to `Image`s. However they can be added to `Kernel`s and then *convolved* with `Image`s (see section 2.3.3) and are useful in a wide range of applications.

```
Calc Avg<Uint8> := [1/9] { 1 1 1, 1 1 1, 1 1 1 };
/* This matrix averages the values in a 3x3 square
        centered on a given pixel */

/* This doesn't work: */
srcimg |= Avg;
```

2.3.3 Kernels

Kernel are ordered collections of Calcs. They are defined with the = (equals) operator and a list of Calcs separated by the — (or) operator. More Calcs can be added to a Kernel afterwards using the |= (or-equal) operator. A Calc in a Kernel can be prefixed with the @ (at) symbol to indicate that it is an intermediate calculation (see section 2.4).

```
Calc sobelGx<Uint8> := {-1 0 +1, -2 0 +2, -1 0 +1};
Calc sobelGy<Uint8> := {+1 +2 +1, 0 0 0, -1 -2 -1};
Calc sobelG<Uint8> :=
    #[sqrt(sobelGx * sobelGx + sobelGy * sobelGy)]#;
Kernel k = | @sobelGx | @sobelGy | sobelG;
/* Calcs can refer to preceding Calcs in same kernel */

Calc sobelTheta := #[atan((float)sobelGx/(float)sobelGy)]#;
k |= sobelTheta;
/* don't have to add all Calcs at once */
```

2.4 Convolutions

The ** operator takes a Channel reference and a Kernel, and applies the Kernel's Calcs in sequence to the Channel. Matrices are applied to the directly to the Image Channel specified, while CStrings generally calculate a value using other Channels. Any Channel which has already been calculated in the convolution may be used by a CString Calc. The result of a convolution is an Image that has an initialized Channel corresponding to each Calc defined in the Kernel which was *not* marked as intermediate (prefixed with @ symbol).

Continuing the previous example, we can take the Kernel, *sobel*, and apply it to an Image:

```
Image edges = srcimg:Lum ** sobel;
/* edges:sobelG and edges:sobelTheta now valid */
/* but not edges:sobelGx or edges:sobelGy */
```

2.5 Full Example

The last few examples have included portions of the *Sobel* edge detecting operator. While in most programming languages implementing the Sobel operator is complicated and error-prone (with multiple nested loops), the CLAM version is straightforward and given in its entirety below:

```
/* Open file given as first command-line argument */
Image srcimg = imgread(1);

/* Define some escaped-C calculations */
Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
Calc sobelG<Uint8>:=
        #[sqrt((float)sobelGx*sobelGx + (float)sobelGy*sobelGy)]#;
Calc sobelTheta<Angle>:= #[atan((float)sobelGy/(float)sobelGx)]#;

srcimg |= Lum;
/* srcimg:Lum created, with values calculated from Red, Green, Blue */

/* The horizontal and vertical gradients at a given pixel */
Calc sobelGx<Uint8> := [1 / 1] { -1  0 +1 ,
                                 -2  0 +2 ,
                                 -1  0 +1 };
Calc sobelGy<Uint8> := [1 / 1] { +1 +2 +1 ,
                                  0  0  0 ,
                                 -1 -2 -1 };

/* Intermediate calculations sobelGx and Gy
   (marked with @) are used to calculate sobelG,
   but don't create their own channels in convolution*/
Kernel sobel = | @sobelGx | @sobelGy | sobelG;

/* sobelTheta also uses the intermediate Calcs Gx and Gy */
sobel |= sobelTheta;

/* Apply sobel to the Lum channel, to get gradient of Luminance */
Image edges = srcimg:Lum ** sobel;
/* edges:sobelG and edges:sobelTheta are defined,
   edges:sobelGx and edges:sobelGy are not */

Image output;
/* An image must have Red, Green and Blue channels
   in order to output properly */
output:Red   = edges:sobelG;
output:Green = edges:sobelG;
output:Blue  = edges:sobelG;

imgwrite( output, "png", 2);
```

Chapter 3

Language Reference Manual

3.1 Introduction

The CLAM programming language is a linear algebra manipulation language specifically targeted for image processing. It provides an efficient way to express complex image manipulation algorithms through compact matrix operations. CLAM programs are first compiled into a "C“ module which is further compiled into a machine binary by an existing C compiler. This two-step process is completely automated by the CLAM compiler, and by default no C code is output (this can be changed with compiler arguments - see section 2.2).

This language reference is inspired by the C reference manual [4]. It details the syntax of the CLAM language.

3.2 Lexical Conventions

3.2.1 Tokens

The tokens in CLAM are broken down as follows: reserved keywords, identifiers, constants, control characters, and operators. The end of a token is defined by the presence of a newline, space, tab character (whitespace), or other character that cannot possibly be part of the current token.

3.2.2 Comments

Comments are demarcated with an opening /* and closing */, as in C. Any characters inside the comment boundaries are ignored. Comments can be nested.

3.2.3 Keywords

The reserved keywords in CLAM are:

Image	imgread	Int8	Uint8
Kernel	imgwrite	Int16	Uint16
Calc	Angle	Int32	Uint32

3.2.4 Identifiers

Identifiers are composed of an upper or lower-case letter immediately followed by any number of additional letters and/or digits. Identifiers are case sensitive, so "foo" and "Foo" are different identifiers. Identifiers cannot be keywords, and cannot start with a digit.

3.2.5 Constants

In CLAM there are 3 types of constants: numeric constants, calculation constants, and string literals.

Numeric Constants

Integers are repesented by a series of number characters.

Angles are represented by a series of number characters with an optional period character.

Calculation Constants

Matrix calculation constants are represented by an opening curly brace, followed by a series of *numeric-expressions* separated by whitespace or comma characters. The comma characters represents the division between the rows of the matrix. Each row must have the same number of *numeric-expressions*, but the matrix need not be square.

A calculation constant may also have an optional fraction preceding it, which indicates that every value in the matrix should be multiplied by that fraction. The fraction will be expressed as an opening bracket character, a *numeric-expression* representing the numerator, a forward-slash character, a *numeric-expression* representing the denominator, and a closing bracket character.

{ *numeric-expr numeric-expr* . . . , *numeric-expr numeric-expr*. . . }
[*numeric-expr* / *numeric-expr*]{ *numeric-expr numeric-expr* . . . , *numeric-expr numeric-expr*. . . }

The following is an example of a calculation constant.

```
Calc sobelGy := [1 / 9]{1 3 1 , 2 -5 2 , 1 3 1 };
```

String Literals

String constants are demarcated by double quote characters or single quote characters. Consecutive string constants will be automatically appended together into a single string constant. String constants may contain escaped characters. Escaped characters begin with a backslash, \, and are followed by either an octal, hexadecimal or base-10 integer value. The following escaped characters are also supported:

`\n` (newline)
`\t` (tab)
`\b` (break)
`\r` (carriage-return)

`"`*string-constant*`"`
`"`*string-constant*`"` `"`*string-constant*`"` ...

3.3 Meaning of Identifiers

3.3.1 Basic Types

There are three basic types defined by the CLAM language. Type identifiers always begin with an upper-case letter followed by a sequence of zero or more legal identifier characters. The list of built-in types is as follows:

```
Image
Calc
Kernel
```

Atom Types

The `Calc` type may be further modified to specify individual element, or "atom" types. This specifies the type of the resulting calculation performed by a `Calc` object (either CString or Matrix – see section 3.4.1). When calculation results exceed the bounds of the specified type, values are clamped (set to the max or min value appropriately). An *atom-type* identifier is denoted using the `<` and `>` characters immediately following the identifier of the object whose atom type is being specified:

`Calc` *identifier*`<`*atom-type*`>`

Legal *atom-type*s are as follows:

Uint8
Uint16
Uint32
Int8
Int16
Int32
Angle

In the absence of an atom-type specification, the atom-type defaults to Uint8, which implies a range of integers from 0-255 inclusive. (This is also the atomic-type for the default *Red*, *Green* and *Blue* channels.)

3.4 Objects and Definitions

An *object* in CLAM is either a named collection of Channels, called an `Image`, or a named collection of calculation bases, called a `Kernel`. A Channel is a mathematical matrix of numeric values whose individual components are not directly accessible via CLAM language semantics – Channel values are manipulated via the convolution operator (see 3.5.5). A calculation basis, known as a `Calc`, is either a calculation constant (see 3.2.5) or a calculation expression (see 3.5.6).

3.4.1 `Calc` objects

A `Calc` object is an immutable object initialized with the `:=` assignment operator (see section 3.5.8). Its assigned value is either a CString (section 3.5.6), or a calculation constant (*matrix* - section 3.2.5). The calculation described by a `Calc` object will be performed for each pixel in an `Image` Channel. The calculation is performed using either the convolution operator (section **??**), or the channel composition operator (section 3.5.8).

3.4.2 `Image` objects

An `Image` is a collection of named Channels. Channels can be dynamically added using the channel composition operator (see section 3.5.8, or by assigning to a previously undeclared Channel name.

For example, to create a gray-scale image from a single, pre-existing Channel:

```
Image outImg;
outImg:Red = calcImg:G;
outImg:Green = calcImg:G;
outImg:Blue = calcImg:G;
```

3.4.3 `Kernel` objects

A `Kernel` is an ordered collection of calculation bases which is used by the convolution operator (see section 3.5.5). Each calculation must to be identified with a `Calc` identifier, but the underlying basis can be either a matrix calculation constant (see 3.2.5) or an escaped C expression (see 3.5.6). A `Kernel` is composed either using the composition operator (see section 3.5.4), or the `|=` assignment operator (see section 3.5.8).

To see how a `Kernel` is used in calculation, see section 3.5.5.

3.5 Expressions

3.5.1 Primary Expressions

CLAM primary expressions can be identifiers, constants, or strings. The type of the expression depends on the identifier, constant or string.

3.5.2 Unary Operators

There are two unary operators in CLAM, and they are only used with a numeric-valued operand such as a numeric constant (see 3.2.5). These expressions are grouped right-to-left:

`+`*unsigned-integer*
`-`*unsigned-integer*

+ operator

This operator forces the value of its numeric operand to be positive. The resulting expression is of numeric type with a value equal to the value of the numeric operand.

- operator

This operator forces the value of its numeric operand to be negative. The resulting expression is of numeric type with a value equal to the negative of the numeric operand.

3.5.3 Channel/Calc Expresions

Channel and `Calc` types are the basis of `Image` and `Kernel` objects respectively. There are several operators that manipulate Channels and `Calcs`.

: operator

Extract, reference or assign an individual Channel in an image.

image-identifier : *channel-identifier*

The resulting expression has a type corresponding to the extracted Channel.

3.5.4 Composition Operators

These operators compose an `Image` or `Kernel` from one or more `Calc` objects. All channel composition operators are left-to-right associative.

`|` operator

Define of list of (one or more) `Calc`s. The resulting expression is a *multi-calc* expression, and can be assigned to a `Kernel` object.

`|` *calc-expression*
multi-calc-expression `|` *calc-expression*

Note that `Calc`s are appended in order, and subsequent operations may rely on this order. Also note that even single `Calc` identifiers must be preceded by `|`

3.5.5 `**` operator

The convlution operator, or `**`, performs the calculations of a `Kernel` on a Channel, and evaluates to a new `Image`. The resulting `Image` will have a Channel for the result of each (non-transient) calculation. When a `Kernel` is evaluated, as many operations as possible are parallelized to improve performance.

Parallelized calculations may depend on each other as long as the dependent calculation is listed after its dependency when the `Kernel` is defined, and the dependent calculation does not depend on more than the current pixel of the required calculation.

Any channels marked with an `@` symbol are transient and are not added to the result `Image`.

3.5.6 Escaped “C” Expression

Escaped “C” expressions, or *CStrings*, are snippits of “C” code which will be run once for every pixel in an `Image` Channel. The string may reference other Channels in the `Image` provided the Channels have been previously defined. To reference a Channel within a CString, the name of the Channel is used just like a local variable in C. CStrings are used on the right side of the `:=` operator when defining a Channel.

The escaped code must be a single expression in C that will evaluate to the type defined by the containing `Calc` object. The expression cannot contain the following characters/tokens: `;` `#` { } `"` `’` `/*` `*/`. Parentheses may be used to group expressions or cast objects, but all parenthesis within the expression must be matched.

`Calc` types and their C-equivalent types:

CLAMtype	equivalent C type
Uint8	uint8_t
Uint16	uint16_t
Uint32	uint32_t
Int8	int8_t
Int16	int16_t
Int32	int32_t
Angle	float

When the channel described by a CString must be evaluated, every pixel value in the channel is calculated by evaluating the C expression. When the expression is evaluated, every identifier corresponding to another channel in the image will be replaced by the value of the pixel in the same location in the referenced channel. Thus, if the C expression contains the identifier `Red`, then when the channel is calculated it will replace `Red` in the expression with the appropriate value from the `Red` channel.

All standard C operators are available for use, as well as any library functions defined in `math.h`. Although bracket characters are not allowed within CStrings, the *ternary* conditional operator, `a ? b : c` is allowed. This enables more complex pixel value calculations such as thresholding and hysteresis.

3.5.7 I/O Expressions

`imgread` expression

The `imgread` expression reads in an `Image` object from a known image format located on the file system. The expression results in an `Image` object which can be assigned using the `=` operator (see section 3.5.8). The resulting `Image` object has 3 Channels named *Red*, *Green*, and *Blue*. Each of the channels correspond to the red, green, and blue image data read into the `Image` object. This expression is invoked as a "C" style function, and expects 1 parameter: either the path of the image file to read (expressed as a string constant); or the number of the command-line argument, an integer 1 or greater.

`imgread(` *string-constant* `|` *integer* `)`

`imgwrite` expression

The `imgwrite` expression writes out an `Image` object to a known image format. It requires that the `Image` object has at least 3 named `Channels`: *Red*, *Green*, and *Blue*. This expression has no type (null type), and is invoked as a "C" style function. It expects 3 parameters: the first parameter is an `Image` identifier, the second is the image format, and the third is the path to which the image should be written (or an integer which represents a command-line argument, as for `imgread`).

`imgwrite(` *image-identifier* `,` *string-constant* `,` *string-constant* `|` *integer* `)`

3.5.8 Assignment Expressions

= assignment operator

Assigns the value of the right operand to the left operand, copying data as necessary. The types of both operands must match. Cannot be used with `Calcs`, which are defined once only (see `:=` below).

The type of this expression is equal to the type of the left operand, and assignment operations may be chained together. For example:

```
Image a;
Image b;
imgwrite(a = b = imgread("foo.jpg"), "png", "foo.png");
```

:= assignment operator

Assigns a calculation constant (see section 3.2.5), or escaped "C" expression (see section 3.5.6) to a `Calc` object. Only used once for each `Calc`, with declaration.

|= assignment operator

Add a single Channel or a (possibly one-member) list of `Calcs` object to an `Image` or `Kernel` object. Assignments using this operator are ordered by statement order, and subsequent operations can rely on this order.

3.6 Statements

Statements in CLAM always end in a semi-colon. No statement can return a value. All statements should either declare a variable, define or modify the definition of a variable, execute some calculation based on previously declared variables with the result stored in previously declared variables, or write an image to a file. "Statements" consisting of a lone r-value (an identifier, channel reference, escaped-C string, matrix, or `imgread()` call) will be accepted but will perform no useful action - they are evaluated but their return values are discarded. Additionally, the CLAM compiler may optionally remove these statements as an optimization.

3.7 Program Definition

A program in the CLAM language is simply a sequence of statements which are executed in order.

3.8 Scope Rules

All identifiers in the CLAM language are global.

In a CString that defines a channel, the existing channels for an image will be in scope when the block is executed. Because this block will be executed on every pixel, the name of the channel will bind to the current pixel value for that channel. These bindings will be resolved when the channel is calculated, not when it is defined, and will be removed after calculation.

3.9 Declarations

All variables must be declared before they can be used. However, variable declarations can be made at any point in a program. A variable becomes usable after the end of the semi-colon of the statement in which it is contained.

3.10 Grammar

3.10.1 Expressions

expression:	*identifier*
	integer
	literal-string
	c-string
	matrix
	matrix-scale matrix
	kernel-calc-list
	channel-ref
	identifier `=` *expression*
	channel-ref `=` *expression*
	channel-ref `**` *identifier*
	identifier `|=` *expression*
	library-function `(` *argument-list* `)`
matrix-scale:	`[` *integer* `/` *integer* `]`
matrix:	{ *row-list* }
	matrix-scale { *row-list* }
row-list:	*matrix-row*
	row-list `,` *matrix-row*
matrix-row:	*integer*
	matrix-row integer
kernel-calc-list:	`@`*identifier*
	`|` *identifier*
	`|` `@`*identifier*
	kernel-calc-list `|` *identifier*
	kernel-calc-list `|` `@`*identifier*
channel-ref:	*identifier*`:`*identifier*
argument-list:	*literal-string*
	argument-list `,` *literal-string*
library-function:	`imgread`
	`imgwrite`

3.10.2 Declarations

declaration:	`Image` *identifier*
	`Kernel` *identifier*
	`Calc` *identifier*
	`Calc` *identifier*`<`*atomic-type*`>`
atomic-type:	`Uint8`
	`Uint16`
	`Uint32`
	`Int8`
	`Int16`
	`Int32`
	`Angle`

3.10.3 Statements and Programs

statement:	*expression* `;`
	declaration `;`
	declaration `=` *expression* `;`
	declaration `:=` *expression* `;`
program:	*statement*
	statement program

3.11 Examples

3.11.1 Gaussian Blur

Figure 3.1 shows an example CLAM program which performs a Gaussian blur on an input image. The input and resulting output images are shown in Figures 3.2a and 3.2b.

```
Image a = imgread(1);
Calc gauss := [1 / 159]
{  2  4  5  4  2,
   4  9 12  9  4,
   5 12 15 12  5,
   4  9 12  9  4,
   2  4  5  4  2
};
/* Create a kernel object for computing the blur */
Kernel dogauss = | gauss;
Image outR = a:Red ** dogauss;
Image outG = a:Green ** dogauss;
Image outB = a:Blue ** dogauss;

Image out;
out:Red = outR:gauss;
out:Green = outG:gauss;
out:Blue = outB:gauss;
imgwrite( out, "png", 2);
```

Figure 3.1: Gaussian Blur Implemented in CLAM

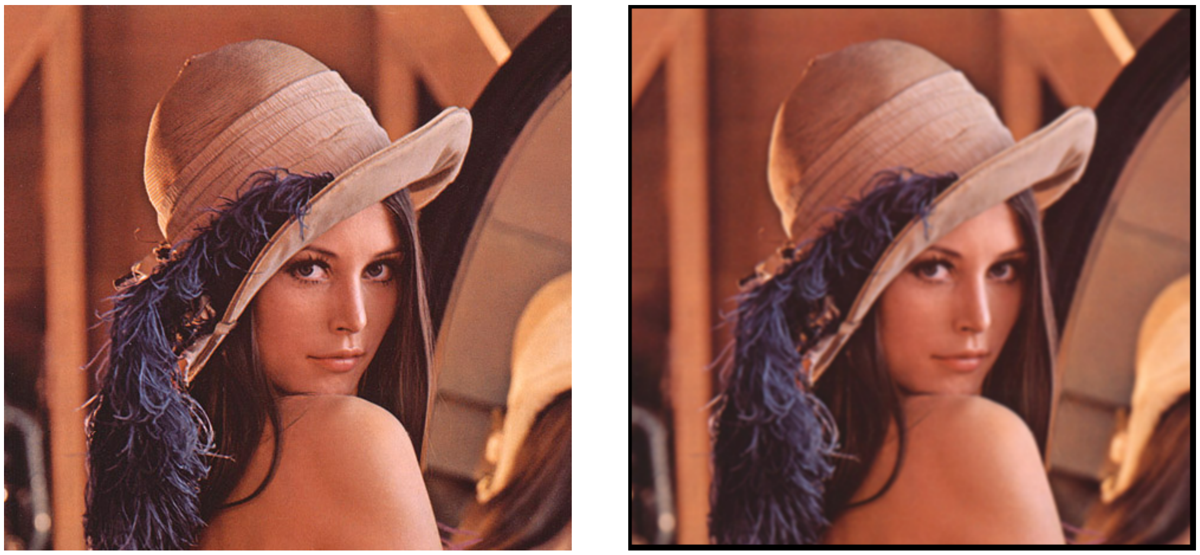

(a) Input Image (b) Blurred Output Image

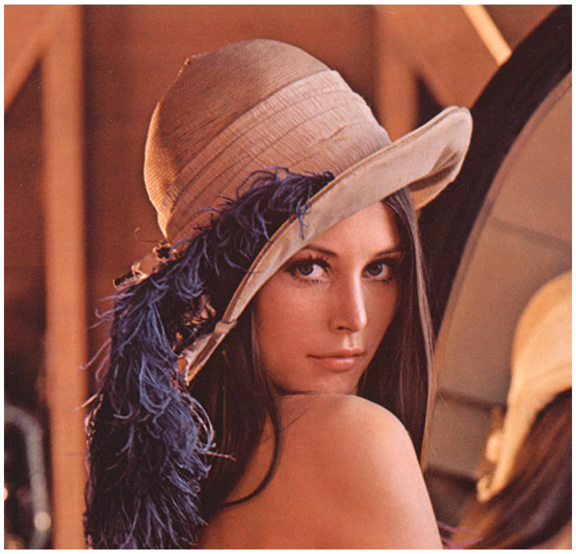
Figure 3.2: Gaussian Blur CLAM Example

3.11.2 Image Segmentation

Figure 3.3 shows an example CLAM program which performs basic image segmentation. Pixels with a luminance value greater than 200 are displayed as red, pixels with a luminance less than or equal to 80 are displayed as blue, and pixels in between are displayed as Green. The input and resulting output images are shown in Figures 3.4a and 3.4b.

```
Image a = imgread(1);

Calc Lum   := #[ (3*Red + 6*Green + Blue)/10 ]#;
Calc highC := #[ Lum > 200 ? 255 : 0 ]#;
Calc midC  := #[ Lum > 80 ? (Lum <= 200 ? 255 : 0) : 0 ]#;
Calc lowC  := #[ Lum <= 80 ? 255 : 0 ]#;

a |= Lum;
a |= highC;
a |= midC;
a |= lowC;

Image out;
out:Red = a:highC;
out:Green = a:midC;
out:Blue = a:lowC;

imgwrite(out, "png", 2);
```

Figure 3.3: Simple Image Segmentation Implemented in CLAM


(a) Input Image

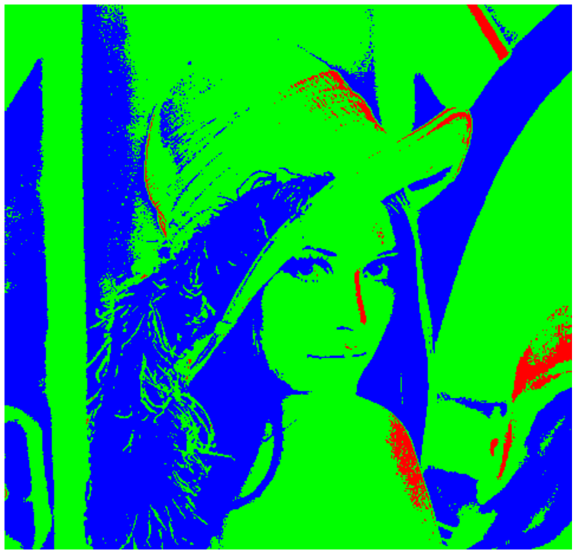

(b) Segmented Output Image

Figure 3.4: Image Segmentation CLAM Example

Chapter 4

Project Plan

4.1 Overview

The CLAM project used a distributed development model as suggested by the `git` [5] version control system. Each team member was expected to keep up with overall development through automated emails sent out via `git` hook scripts run on every successful push.

Initial planning and design of the CLAM language came out of the previous experience of group members in image processing. Specification of language syntax was driven by the desire to simplify basic image processing and linear algebra calculations which are onerous to implement in other languages. The core of this development hinged on the *convolution* (section 3.5.5) operator. When used in combination with a flexible matrix definition syntax, a simple convolution operator can eliminate quadruply–nested for–loops and clarify program operation. This paradigm lead us directly to our elegant matrix definition syntax, and a `Kernel` (section 3.4.3) formalism that allows efficient ordering of calculations and further reduces the number of lines of code necessary to implement most algorithm.

We took a practical approach to formalizing the language semantics by attempting to implement a real–world algorithm in the most compact yet readable manner possible. We chose the *Sobel* [6] operator which is a standard edge detection algorithm used in computer vision. Calculating the Sobel operator involves a luminance calculation, two separate convolutions, and two additional calculations on the convolution output. Using this algorithm as a guide, we developed a syntax which balanced complexity and readability, and is able to implement the complete Sobel operator in 12 lines of code (see section 2.5 for a source listing).

After language specification and semantics were reasonably well-defined, we began development of the scanner and parser. We quickly realized that the most challenging pieces of the CLAM design would be the strict type–checking and backend C implementations. With this in mind, the project was put on a loose timeline where priority was given to the verifier (see section 5) and to the C library implementation of the CLAM syntax.

Fortunately formal verification of types and syntax could happen substantially in parallel with the backend and semantically checked abstract syntax tree (SAST) generation. However, we found that the details of the generated C code, the backend implementation, and the SAST were inexorably entwined. Thus the major focus of development at the end of the project was on this area, and collaborative programming techniques such as pair-programming were employed to speed development and improve code quality.

Our testing procedure, described in chapter 6, is based around unit tests divided into four major categories: syntactic tests, semantic/type tests, CString tests, and functional output tests. Each category is designed to exercise a specific area of the CLAM compiler, and each test uses a simple pass/fail metric. Unit tests were developed in parallel to other development, and were used extensively to fix bugs and refine the language syntax.

4.2 Administration

Although CLAM development did not follow a strict timeline, there was a well-defined order in which major pieces of the design had to be completed. Figure 4.1 shows the timeline that CLAM development followed which was reconstructed from the version control logs. Note that there was a compression of activity near the end of the development cycle. This was an unfortunate side-effect of a loosely-defined schedule.

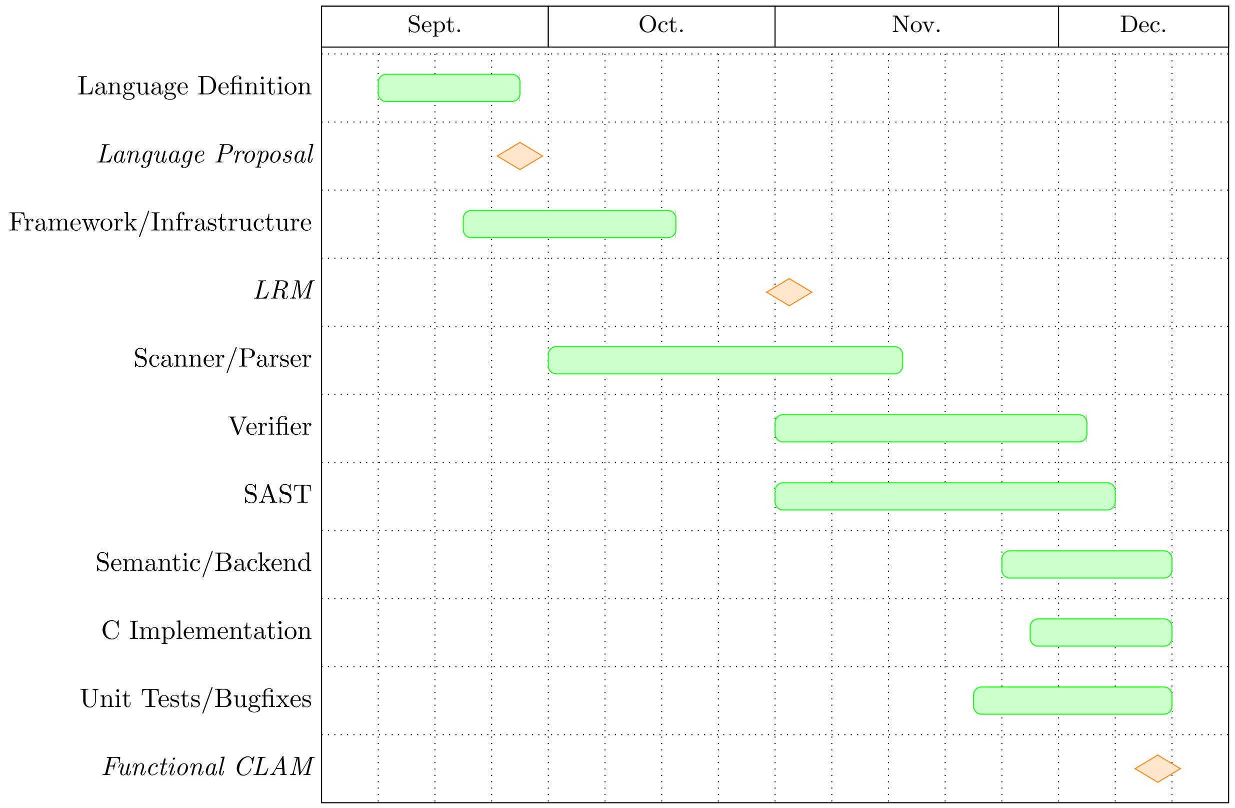


Figure 4.1: CLAM Development Timeline

Each member of the CLAM team played a role in the development process. Figure 4.2 details the code and documentation responsibilities of each member. In addition to each documentation area specified in the figure, each team member contributed a *lessons learned* paragraph. These were collected into Chapter 7.

Developer	**Code Responsibilities**	**Documentation Responsibilities**
Jeremy Andrus	Framework, Scanner/Parser, Verifier, C Implementation	Introduction (Ch.1), LRM (Ch.3), Project Plan (Ch.4), Code Reference (App.A), VCS History (App.B)
Robert Martin	SAST, Semantic, Backend	Architectural Design (Ch.5), LRM (Ch.3), Introduction (Ch.1)
Kevin Sun	Unit Tests, Verifier Bugfixes	Language Tutorial (Ch.2), Test Plan (Ch.6), LRM (Ch.3)
Yongxu Zhang	Unit Tests, Verifier Bugfixes	Test Plan (Ch.6)

Figure 4.2: Roles and Responsibilities

4.3 Development Environment

The CLAM compiler is written in Objective Caml (OCaml) and C/C++. The bulk of the compilation is done in OCaml which generates C/C++. A CLAM C library was written which implements the low-level details of the image processing described by the language including structures, memory management, and a convolution framework written in a C++ template function.

Compiling CLAM requires the `gcc` compiler and the OCaml toolchain available for download from the OCaml website: `http://caml.inria.fr/download`. In order to produce machine-executable binaries, CLAM must invoke the `g++` compiler which should be accessible from the user's default *PATH*.

CLAM documentation, including all charts, code examples, and course-related documents, were written in LaTeX. This allowed the team to work on the same document in a coherent, distributed manner.

All source code and documentation was managed in the `git` [5] distributed version control system. A "master" repository was kept on a Columbia CRF backed-up server, and *hook* scripts were used to notify all team members when code was pushed to this repository. A complete log of project activity can be found in Appendix B.

Chapter 5

Architectural Design

Our design incorporates six layers: the scanner, the parser, the verifier, the semantic converter, the backend C generation, and C image manipulation libraries (see Figure 5.1). The compiler takes as input a text file containing a valid CLAM program and outputs a binary that implements the program described by the input text.

The scanner generates as its output a one-dimensional list of tokens (see Figure 5.2).

The parser reads this token list and uses our context-free grammar to generate an abstract syntax tree. The parser is not strict about which nodes have which children, or whether or not it is meaningful to have the current structure: rather it just blindly assembles the tree. The abstract syntax tree has the nodes described in Figure 5.3.

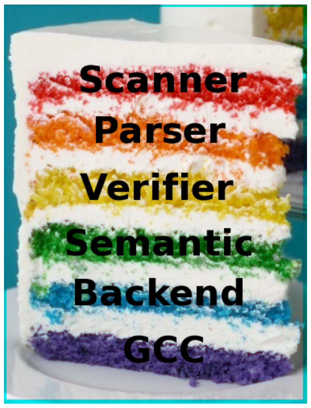


Figure 5.1: A diagram showing the layers of CLAM. The raw source code is fed into the top of the cake and binary executables come out the bottom. (NOTE: This image was manipulated by CLAM using a Gaussian blur filter.)

The verifier accepts as input an abstract syntax tree. It traverses the tree and checks that nodes are arranged in meaningful ways. While it traverses, it also builds up an environment that keeps track of all variables defined, along with their identifier name and type. If the verifier accepts the AST, it will return as output the cumulative environment that contains the identifiers and their types.

SEMI	LPAREN	RPAREN	LTCHAR	GTCHAR	LBRKT	RBRKT
LBRACE	RBRACE	COLON	COMMA	FSLASH	CONVOP	PIPE
ATSYM	UMINUS	UPLUS	ASSIGN	DEFINE	OREQUAL	IMAGET
KERNELT	CALCT	CHANNELT	UINT8T	UINT16T	UINT32T	INT8T
INT16T	INT32T	ANGLET	IMGREAD	IMGWRITE	LITSTR	CSTR
INTEGER	ID	EOF				

Figure 5.2: A comprehensive list of tokens produced by the lexical analysis (scanner).

Node	**Description**
stmt	Statement
vdecl	Variable Declaration
expr	Expression
matrix	Calc matrix
bareint	Integer
kerncalc	Kernel calculation
chanref	Image channel reference
libfunc	CLAM library function
assign_op	Assignment operation
atom	Atomic type

Figure 5.3: List of Abstract Syntax Tree Nodes.

Layer	**Lead Developer**
Unit Tests	Kevin and Yongxu
Scanner	Jeremy
Parser	Jeremy
Verifier	Jeremy / Robert
Semantic	Robert
Backend	Robert / Jeremy
GCC/G++	Richard Stallman et al.

Figure 5.4: Lead Developer of Each Layer.

The semantic converter takes in a verified abstract syntax tree and also the list of variables generated by the verifier. It then maps each node or configuration of nodes to a corresponding Semantic AST node that corresponds more directly with the eventual C code that must be generated. The output of this layer is a semantically checked abstract syntax tree (SAST). The semantic converter also supplements the environment information inherited from the verifier with additional information not related to verification, such the largest referenced command-line argument.

The backend takes as input the SAST and the supplemented environment information. It then converts this into a meaningful C++ program source file. This program can be submitted to GCC and compiled.

The final layer of CLAM is the GCC compiler. The generated C source from the backend is fed into the GCC compiler, which outputs a binary in architecture-specific assembly language.

The lead developers for each layer of CLAM are identified in Figure 5.4.

Chapter 6

Test Plan

Our unit test framework consists of pairs of identically named files in the `clam/tests/` directory. Each pair consists of a CLAM program with extension `.clam` and an executable shell script with extension `.test`. The CLAM file contains the code to be tested, while the shell script specifies how to test that code: whether to compile and run or only compile, whether the test is supposed to fail, what the expected output should look like, and what command line arguments to pass. The `_buildup.sh` sets up the testing environment and defines common procedures such as `compile-it` and `run-it`. Furthermore, `all.test` runs all tests in the directory, tallies successes and failures and outputs a summary at the end.

Our testing is divided into four sections: syntax verification (section 6.1), semantic/type verification (section 6.2), CString verification (section 6.3), and functional output verification (section 6.5) which tests image processing results by comparison.

6.1 Syntax Verification

Syntax verification testing is meant to confirm that the parser accepts all valid token strings, and rejects all invalid ones as defined in our language reference manual. We achieve this by inspecting `clam/parser.mly` and writing unit tests for potentially problematic cases (many of the more straightforward rules were not deemed test-worthy). All of these tests are only compiled and not executed. The testing process uncovered a number of errors in matrix parsing and definition of kernel calculation lists.

- `matrix1.clam` tests that a simple matrix parses correctly, and should compile:

```
Calc mat := { 0 0 0 , 0 0 0 , 0 0 0 };
```

  (This originally failed because a scale factor was required, but now it is accepted.)

- `matrix2.clam` tests that a matrix with scaling factor parses correctly, and should compile:

```
Calc mat := [1 / 1] { 1 1 1 , 1 1 , 1 1 1 1 };
```

- `matrix3.clam` tests whether the parser catches ill-formed matrices, and should not compile:

```
Calc m := { 0 0 0 , 0 0 0 ,;
```

(This originally succeeded due to incorrect matrix parsing rules, but now it fails.)

- `matrix4.clam` tests whether the parser catches another type of ill-formed matrix, and should also not compile:

```
Calc m := { 0 0 0 } 0 0 0 } 0 0 0 };
```

(This originally succeeded due to incorrect matrix parsing rules, but now it fails.)

- `keyword-id.clam` tests whether the parser allows keywords to be identifiers, and should not compile:

```
Calc Kernel := [1/1]{1 1 1, 1 1 1, 1 1 1};
```

- `1calc-ker.clam` tests whether the parser allows `Kernel` definitions with only one `Calc`, and should compile:

```
Calc Kernel := [1/1]{1 1 1, 1 1 1, 1 1 1};
```

(This originally failed because the original syntax caused reduce/reduce errors, so we changed both the parser and the test. There was originally no `|` after the `=`.)

- `string1.clam` checks that consecutive string literals are concatenated together into a single string literal. This should compile:

```
Image a = imgread("foo" "bar" ".jpg");
```

- `convoperand.clam` tests whether the `**` enforces a strict order of operands, and should not compile:

```
Image img;

Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | @sobelGx | @sobelGy | sobelG;

Image img2 = sobel ** img:Red;
```

(We only allow convolutions of the form *channel-ref* `**` *identifier*, in order to simplify to translation into C code.)

- `defcalc1.clam`, `defcalc2.clam`, and `defcalc3.clam` test various ways of declaring `Calc`s, and all three should compile:

```
Calc id<Uint8> := [1 / 1] { 1 0 0 , 0 1 0 , 0 0 1 };
```

```
Calc id<Uint8>;
```

```
Calc id<Uint8>;

id := [1 / 1] { 1 0 0 , 0 1 0 , 0 0 1 };
```

- `rval-calc.clam`, `rval-matrix.clam`, `rval-chanref.clam`, `rval-conv.clam`, `rval-cstr.clam`, `rval-image.clam`, `rval-imgread.clam`, and `rval-kernel.clam` test various ”do nothing” statements consisting solely of r-values. Theses should all compile, though their return values of the last line of each test are discarded so they do nothing:

```
Calc c := { 1 0 , 0 1 };
c;
```

```
{ 0 1 2 , 3 4 5 , 6 7 8 };
```

```
Image srcimg = imgread("ucla.png");

srcimg:Red;
```

```
Image srcimg = imgread("ucla.png");

Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
Calc sobelG<Uint8>:= #[sqrt(sobelGx*sobelGx + sobelGy*sobelGy)]#;

srcimg |= Lum;

Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | @sobelGx | @sobelGy | sobelG;

srcimg:Lum ** sobel;
```

```
#[Red + Green + Blue]#;
```

```
Image s = imgread("ucla.jpg");
s;
```

```
imgread("ucla.jpg");
```

```
Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | sobelGx | sobelGy;

sobel;
```

(The parser accepted all of these, though most originally caused errors in the C translator that had to be fixed (see ”C Compiler Verification” below).)

- `equality-trans.clam` checks that assignment expressions can be nested. This should compile:

```
Image a;
Image b;
Image src = imgread("ucla.png");

a = b = src;

a:Red;
```

- `comment1.clam` checks that a program with only a comment (and zero statements) compiles. This should compile:

```
/* This is a sample comment. */
```

6.2 Semantic Verification

Semantic verification testing is meant to confirm that the verifier accepts all valid parse trees, and rejects all invalid ones according to the specifications of our language (and according to what makes sense). These tests depend on `clam/verifier.ml`, as well as `clam/environ.ml` and `envtypes.mli`. The testing process uncovered a number of errors in matrix verification (separate from syntax) and the creation of default RGB Channels.

- `zerocalc1.clam` and `zerocalc2.clam` check that a matrix scaling-factor can have a numerator of zero, but not a denominator of zero. The former should certainly compile, while the latter should not:

```
Calc id<Uint8> := [0 / 1] { 1 0 0 , 0 1 0 , 0 0 1 };
```

```
Calc id<Uint8> := [1 / 0] { 1 0 0 , 0 1 0 , 0 0 1 };
```

  (`zerocalc2.clam` originally passed verification, but that was fixed.)

- `invalid1.clam` checks that undeclared identifiers are caught. This should not compile:

```
InvalidIdentifier;
```

- `undefined1.clam` checks that an undefined Channel reference cannot be an r-value. This should not compile:

```
Image srcimg = imgread("ucla.png");

Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
srcimg |= Lum;

Image output;
output:foo = srcimg:foo; /* srcimg:foo doesn't exist! */
```

- `imgchannel1.clam` checks that an `Image` has default channels when read. This should compile:

```
Image s = imgread("ucla.png");
Image t;
t:Red = s:Green;
```

- `imgchannel2.clam` checks that an undefined `Calc` cannot be applied to an `Image`. This should not compile:

```
Image s;
Calc C;
s |= C;
```

- `imgchannel3.clam` checks that a `Calc` defined as a matrix cannot be applied to an `Image`. This should not compile:

```
Image s;
Calc C := [1/1] { 0 0 0 , 0 1 0 , 0 0 0 };
s |= C;
```

- `imgchannel4.clam` checks that a `Calc` defined as a CString *can* be applied to an `Image`. This should compile:

```
Image s = imgread("ucla.png");
Calc C := #[Red + Blue]#;
s |= C;
```

- `image-eq-image.clam` checks `=` assignment for images to images. This should compile:

```
Image s = imgread("ucla.png");
Image t = s;
```

- `image-oreq-image.clam` checks `|=` assignment for images to images. This is not supported and doesn't compile:

```
Image s = imgread("ucla.png");
Image t = imgread("ucla2.png");
t |= s;
```

- `image-defeq.clam` checks `:=` assignment for images. This is not supported and doesn't compile:

```
Image s := imgread("ucla.jpg");
```

- `imgread-bad.clam` checks that l-value identifiers have to be declared first. This shouldn't compile:

```
s = imgread("ucla.jpg");
Image s;
```

- `imgread-bad2.clam` checks that `imgread` must be called with only one argument. This shouldn't compile:

```
imgread("ucla.png","ucla2.png");
```

- `imgread-bad3.clam` checks that `imgread` can only be called with a literal string or integer. This shouldn't compile:

```
imgread({ 0 0 0, 0 0 0, 0 0 0 });
```

- `imgread.clam` checks that `imgread` can actually called with a correct argument. This should compile:

```
s = imgread("ucla.jpg");
Image s;
```

- `imgwrite-bad1.clam` checks that `imgwrite` can't be called with incorrect number and type of arguments. This shouldn't compile:

```
imgwrite({ 0 0 0, 0 0 0, 0 0 0 });
```

- `defchannels.clam` checks that the default RGB Channels don't exist for a declared-but-not-read `Image`. This shouldn't compile:

```
Image a;
a:Red; /* should fail */
```

(Allowing default RGB channels for an unread `Image` is problematic because CLAM does not allow explicit manipulation of Channel *size*.)

- `imgwrite-norgb.clam` checks that a declared-but-not-read `Image` without RGB channels (and more importantly, without a *size*) cannot be written. This shouldn't compile:

```
Image a = imgread("ucla.png");
Calc c := #[1]#;
Calc d := #[2]#;
Kernel k = | c | d;
Image b = a:Red ** k;
imgwrite(b, "png", "fail.png");
```

- `sizediff.clam` checks that the default Channels from `Image`s of different sizes cannot be assigned to each other, because all Channels of an `Image` should have the same size. This shouldn't compile:

```
Image a = imgread("flatiron.jpg");
Image b = imgread("ucla.png");
a:Red = b:Red;  /* Should fail because of size difference*/
```

- `at-channel.clam` checks that transient `Calc`s marked with `@` in a `Kernel` do not result in Channels after a convolution. This shouldn't compile:

```
Image srcimg = imgread("ucla.png");

Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
Calc sobelG<Uint8>:= #[sqrt(sobelGx*sobelGx + sobelGy*sobelGy)]#;
Calc sobelTheta<Angle>:= #[arctan(sobelGy/sobelGx)]#;

srcimg |= Lum;

Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | @sobelGx | @sobelGy | sobelG;
sobel |= sobelTheta;

Image edges = srcimg:Lum ** sobel;
Image output;
output:Red   = edges:sobelGx; /* This should be bad */
```

- `addker.clam` checks a `Kernel` can be appended to another `Kernel` using `|=`. This should compile:

```
Image imtest;

Calc c1<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc c2<Uint8> :=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel k1 = | c1 | c2;
Kernel k2 = | c2 | c1;

k1 |= k2;
```

- `DefEq.clam` checks that `=` cannot be used to define a `Calc`. This should not compile:

```
Calc test1<Uint8> = [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
```

- `ckernel.clam` checks that `Kernels` must be defined using `Calc` *identifiers* and not anonymous `Calc` expressions. This should not compile:

```
Kernel t1 = #[Red*2]# | #[Blue]#;
```

- `cimage.clam` checks that `Images` cannot be defined using `Calc` expressions. This should not compile:

```
/* should not allow this: */
Image img = #[23]#;
```

6.3 CString Verification

CString verification testing checks that invalid CStrings are caught before being passed to GCC, where they could cause fatal errors. Of course, it is impossible to catch *all* possible errors without building an understanding of C syntax directly into CLAM, so we are satisfied with catching common mistakes and rely on the user and GCC error messages for everything else. (For example, a CString of the form `#[while(1)]#`, while certainly disruptive, will compile and it is the user's fault that the program stalls.) For each possible issue that we *did* choose to catch, we wrote unit tests to confirm that they were caught. Catching of invalid CStrings is done entirely in `clam/scanner.mll`. Testing reveals errors in parentheses matching, which were promptly corrected.

- `cstring1.clam` checks that C preprocessor cannot be used in a CString. This should not compile:

```
Calc c := #[#ifdef CALC]#;
```

- `cstring2.clam` checks that C comments cannot be used in a CString. This should not compile:

```
Calc c := #[//remove trailing semicolon!]#;
```

- `cstring3.clam` checks that empty CStrings are OK, and just result in an empty statement. This should compile:

```
Calc c := #[]#;
```

- `cstring4.clam` checks that library calls in a CString must be closed. This should not compile:

```
Calc c := #[sqrt(2]#;
```

  (This originally did compile because the scanner didn't record nesting-level properly. It was fixed.)

- `cstring5.clam` checks that parentheses in a CString must be matched. This should not compile:

```
Calc c := #[2*(3-(4+5)]#; /*mismatched parentheses*/
```

  (This originally did compile because the scanner didn't record nesting-level properly. It was fixed.)

- `cstring6.clam` is a sanity check that ”normal” CStrings do in fact work. This should obviously compile:

```
Calc c := #[1+2+3]#;
```

6.4 C Compiler Verification

A few of the above tests, which originally compiled, began failing once the C backend was hooked up. The most common errors were with C syntax and memory allocation. While we didn't write tests specifically targeted at the C backend, the tests we wrote for other parts of the architecture also helped us identify problems in the C translator.

6.5 Image Processing Verification

These are comprehensive tests, consisting of full-fledged programs that actually read, manipulate and write images. While the testing framework provides image-comparison functionality, most verification here is actually done with the naked eye.

- `imgcopy.clam` copies an image. This should compile, run, and copy an image:

```

Image img;
img = imgread("ucla.png");
imgwrite(img, "png", "ucla_out.png");
```

- `sobel.clam` applies the Sobel operation to an image. This should compile, run, and output a map of the luminosity gradient:

```
Image srcimg = imgread(1);

Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
Calc sobelG<Uint8>:= #[sqrt((float)sobelGx*sobelGx + (float)sobelGy*sobelGy)]#;
Calc sobelTheta<Angle>:= #[atan((float)sobelGy/(float)sobelGx)]#;

srcimg |= Lum;

Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | @sobelGx | @sobelGy | sobelG;
sobel |= sobelTheta;

Image edges = srcimg:Lum ** sobel;
Image output;
output:Red   = edges:sobelG;
output:Green = edges:sobelG;
output:Blue  = edges:sobelG;

imgwrite( output, "png", 2);
```

Chapter 7

Lessons Learned

7.1 Jeremy Andrus

Through the course of this project, there are several things that I learned. First, I learned that writing a compiler is much more complicated than I had first imagined. While OCaml's syntax and scanner/parser integration ease development quite a bit, there was still a lot of code to write. Second, I learned that the implementation of a backend generator, the definition of a C API that the generator will use, and the syntax tree that the generator takes as input are all intensely entwined. It was nearly impossible for us to separate the design of the C/C++ implementation library from the generator backend, and similarly it was impossible to separate the backend generator design from the abstract syntax tree that it took as input.

Finally, I learned that self-motivation in a group setting seems to decrease proportionally to the number of members in the group. Idealistic dreams of independantly motivated developers all hacking towards a common goal are naive, and more realistic approaches to group dynamics must be applied. This means that nagging emails and explicit time spent on planning and organizing will inevitably save late-night hacking cycles near the end of the project.

7.2 Robert Martin

It's the little things that end up being the big things. When we started the project, we had a good idea of the functionality we wanted to express in our language, but we did not precisely work out the gritty details of the entire translation from CLAM source code to program execution. We would have benefited greatly by each writing 1 or 2 example programs before we had ever written a line of code. We could then have reconvened to talk through our example programs, discussing at a high level what we would expect the resultant binary to do at each line and the deductive logic that would allow it to do so. Such a conversation at an early stage would have exposed a number of deficiencies and redundancies in the language, and led to a better finished product.

7.3 Kevin Sun

- Make sure you thoroughly understand the Ocaml examples from class.

- Stay in contact with your team throughout the semester, to make sure everyone is always up to speed.
- Define a clear division of labor early, so work is distributed evenly.
- Do lots of testing, even (especially) for apparently simple cases. It's so easy to overlook minor details, and a comprehensive set of tests is a great way to ensure you don't break anything.
- At the same time, definitely try to push your language in testing. Thinking of ways a coder could break your language forces you to think about the details of your implementation.

7.4 Yongxu Zhang

Learned in detail how the AST, scanner, parser and verifier are worked together to translate a programming language into an executable. Under some circumstances, it's not as easy as I thought. For example, one needs to take the precedence and data types check into consideration from very beginning, such as designing a syntax tree. The way in which one part is designed may have significant influence to other parts. When we testing and fixing bugs in syntax, after a lot of syntax errors involving data types were discovered, we rewrote the syntax tree to be strongly typed instead of directly go to modify the parser and verifier, and that saved us a lot of time.

Also I learned how to design unit tests to discover all kinds of dark corners of a language and how to fix them. One cannot follow the routine when designing unit tests. Unusual conditions also need to be considered since they are always useful to detect bugs that out of one's expectation. By retrieving back to the source code with results of other relative tests, the error is located and fixed.

Appendix A

Complete Code Reference

The CLAM programming language was implemented in OCaml. It uses the C/C++ programming languages as backend targets, and invokes the `g++` compiler on the CLAM compilation output to produce a final binary. The final binary is statically linked against a CLAM implementation library, `clam.a`, which contains the low-level image manipulation functions. Additionally, CLAM leveraged several OCaml functions from the *extlib* [3] project, and a stand-alone image reader [1] and writer [2] library written by Sean Barrett. Code listings of the CLAM compiler and CLAM implementation library as well as the subset of *extlib* used by CLAM follow:

A.1 CLAM Compiler

```
(*
 * File: clam.ml
 * Date: 2011-10-16
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

let clamversion = "0.1"
let clam_binout = ref "a.out"
let clam_c_out  = ref "clam_gen.c"
let clam_c_only = ref false
let clam_print_ast = ref false
let clam_srcin  = ref "-"

let clam_usage =
  let s1 = "CLAM v"^clamversion^"\n" in
  let s2 = Printf.sprintf "Usage: %s {options} [{<} inputfile]\n"
           (Filename.basename Sys.executable_name) in
  let s3 = "Options are:" in
  s1 ^ s2 ^ s3

let set_clam_output s = clam_binout := s; clam_c_out := s

let set_clam_input s = clam_srcin := s

let set_clam_gen_c_only () = clam_c_only := true

let set_clam_print_ast () = clam_print_ast := true

let clam_anon_fcn = function
  | "-" -> clam_srcin := "-"
  | filename -> clam_srcin := filename

let _ =
  let args =
    [  "-o", Arg.String set_clam_output, "<filename> Specify the output file";
       "-i", Arg.String set_clam_input, "<filename> Specify the input file";
       "-c", Arg.Unit set_clam_gen_c_only, " Output generated C only";
       "-t", Arg.Unit set_clam_print_ast, " Print AST debugging information";
    ] in
  Arg.parse (Arg.align args) clam_anon_fcn clam_usage;
  try
    let input_prog = if clam_srcin = ref "-" then
                       Parseutil.parse_stdin ()
                     else
                       Parseutil.parse_file !clam_srcin in
    let program = List.rev input_prog in
    (* print out the AST if requested: before verification == debugging :-) *)
    let _ = if !clam_print_ast then
              print_endline (Printer.string_of_ast program) else () in
    let (env, verified_ast) = Verifier.verify program in
    let (scope, sast) = Semantic.translate_ast env verified_ast in
    let c_code = Backend.generate_c scope sast in
    if !clam_c_only then
      let ochan = Pervasives.open_out !clam_c_out in
      let _ = Pervasives.output_string ochan c_code in
      let _ = Pervasives.close_out ochan in
      exit 0
    else
      Clamsys.compile_c c_code !clam_binout; exit 0
  with
      Failure(s)            -> prerr_endline ("Error: "^s); exit 1
    | Semantic.SemanticFailure(s) -> prerr_endline ("Semantic Error: "^s); exit 1
    | Parseutil.ParseErr(e,s) as err -> Printer.print_clamerr err; exit 1
    | Sys_error(s)          -> prerr_endline
                               ("System error - check permissions on '"^
                                 Filename.temp_dir_name^"': "^s); exit 1
    | Clamsys.Compile_error(s) -> prerr_endline ("C-Backend error: "^s); exit 1
```

```
(*
 * File: ast.mli
 * Date: 2011-10-10
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

type atom = Uint8 | Uint16 | Uint32 | Int8 | Int16 | Int32 | Angle | Unknown
type assign_op = Eq | OrEq | DefEq

type libfunc = ImgRead | ImgWrite

type chanref = { image : string; channel : string; }

type kerncalc = { allcalc: string list; unusedcalc: string list }

type bareint =
    BInt of int

type matrix = (bareint * bareint) * bareint list list

type expr =
    Id of string
  | Integer of bareint
  | LitStr of string
  | CStr of string * string list
  | KernCalc of kerncalc
  | ChanMat of matrix
  | ChanRef of chanref
  | Convolve of chanref * string
  | Assign of string * assign_op * expr
  | ChanAssign of chanref * expr
  | LibCall of libfunc * expr list

type vdecl =
    ImageT of string
  | KernelT of string
  | KCalcT of kerncalc (* need this to keep used/unused list around! *)
  | ConvT of chanref * string
  | CalcT of string * atom
  | StrT of string * string
  | BareT of string

type stmt =
    Expr of expr
  | VDecl of vdecl
  | VAssign of vdecl * assign_op * expr

type program = stmt list
```

```
(*
 * File: backend.ml
 * Date: 2011-10-17
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

open String
open Envtypes
open Ast
open Sast

(*
 * Identifier Translations
 *)
let id_of_imgId imgId = "__imgT_" ^ imgId
let id_of_kernId kernId = "__kernT_" ^ kernId
let id_of_calcId calcId = "__calcT_" ^ calcId

let id_of_imgT imgT = id_of_imgId imgT.iname
let id_of_kernT kernT = id_of_kernId kernT.kname
let id_of_calcT calcT = id_of_calcId calcT.cname
let id_of_chanT chanT = "clam_imgchan_ref( " ^
                        (id_of_imgId(fst chanT)) ^ ", \"" ^
                        (escaped(snd chanT)) ^ "\")"

(*
 * Variable Declarations
 *)
let c_of_imgDecl imgT = "clam_img *" ^ (id_of_imgT imgT) ^ " = NULL;\n"
let c_of_kernDecl kernT = "clam_kernel *" ^ (id_of_kernT kernT) ^ " = NULL;\n"
let c_of_calcDecl calcT = "clam_calc *" ^ (id_of_calcT calcT) ^ " = NULL;\n"

(*
 * Variable Definitions
 *)

(* TODO: Initialize the Calc variables? *)


(*
 * Main C Functions
 *)

let c_of_atom = function
    Uint8  -> ("UINT8", "uint8_t")
  | Uint16 -> ("UINT16", "uint16_t")
  | Uint32 -> ("UINT32", "uint32_t")
  | Int8   -> ("INT8", "int8_t")
  | Int16  -> ("INT16", "int16_t")
  | Int32  -> ("INT32", "int32_t")
  | Angle  -> ("ANGLE", "float")
  | _ -> raise(Failure("Backend finding the Calc type of Unknown or Angle?"))

let c_of_fmt = function
    Png -> "PNG"
  | Bmp -> "BMP"
  | Tga -> "TGA"


(* Generate the #define CHANNEL (VALUE) statements for a convolution
 * based on channel identifiers parsed out of the literal C string
 *)
let inner_c_of_cfunc_calc calclst ct =
  let cinfo_from_id id = (List.fold_left (fun (ii,idx,ct) c ->
                                          if (c.cname = id) then (ii+1,ii,c) else (ii+1,idx,ct))
                          (0,0,ct)
                          calclst) in
  let pixparms ct idx = (snd (c_of_atom ct.ctype))^","^(string_of_int idx) in
  let cdef_of_cfunc_id id =
    let _,idx,idcalc = cinfo_from_id id in
    "\t\t#define "^id^"  clam_img_pix("^(pixparms idcalc idx)^")\n"
  in
  let idlist = (snd ct.cfunc) in
  (List.fold_left (^) "" (List.map cdef_of_cfunc_id idlist)) ^
  "\t\t#define cfunc ("^(fst ct.cfunc)^")\n"^
  "\t\tclam_convolve_cfunc("^(id_of_calcId ct.cname)^","^
                          (snd (c_of_atom ct.ctype))^",cfunc)\n"^
  "\t\t#undef cfunc\n"^
  (List.fold_left (^) "" (List.map (fun x -> "\t\t#undef "^x^"\n") idlist))

let c_of_cfunc_calc calclst ct = if (ct.cismat) then "" else
  "\tdo_"^ct.cname^":\n" ^
  (inner_c_of_cfunc_calc calclst ct)^
  "\t\tcontinue;\n"

let c_of_convData cvdata =
  let kernId, chanref, calclst, idx = cvdata in
  let chk_wrapper ct = if ct.cismat then "" else "\tclam_convfunc_chk("^ct.cname^")\n" in
  "\nclam_convfunc_start("^(string_of_int idx)^","^
                        (id_of_imgId (fst chanref))^","^(snd chanref)^")\n"^
  (List.fold_left (^) "" (List.map chk_wrapper calclst))^
  "\t\tclam_convfunc_lastchk()\n"^
  (List.fold_left (^) "" (List.map (c_of_cfunc_calc calclst) calclst))^
  "clam_convfunc_end("^(string_of_int idx)^")"

let c_of_fid = function
    Arg(i) -> "argv[" ^ (string_of_int i) ^ "]"
  | Const(s) -> "\"" ^ (escaped s) ^ "\""

let c_of_matrix m =
  let ( (wid, hei), (num, den), rowCol ) = m in
  (string_of_int wid) ^ ", " ^ (string_of_int hei) ^ ", " ^
  (string_of_int num) ^ ", " ^ (string_of_int den) ^ ", " ^
  let c_of_matrix_row int_row =
    let str_row = List.map string_of_int int_row in
    let rs = (List.hd str_row) :: (List.map ((^) ", ") (List.tl str_row)) in
    "{" ^ (List.fold_left (^) "" rs) ^ "}"
  in
  let raw_rows = List.map c_of_matrix_row rowCol in
  let rows = (List.hd raw_rows) :: (List.map ((^) ", ") (List.tl raw_rows)) in
  "{ " ^ (List.fold_left (^) "" rows) ^ " }"

let is_kernel_used id unused = if (List.exists (fun c -> c = id) unused) then "0" else "1"

let rec c_of_kernCalc assignTo unused = function
    [] -> (if assignTo = "" then "clam_kernel_alloc()" else (id_of_kernId assignTo))
  | hd :: tl -> "clam_kernel_addcalc("^(c_of_kernCalc assignTo unused tl)^","^
                                     (id_of_calcId hd)^","^
                                     (is_kernel_used hd unused)^")"


let c_of_declare_matrix cid t mat =
  let ident = id_of_calcId cid in
  let (bigSz, smallSz) = c_of_atom t in
  ident ^ " = clam_calc_alloc(\"" ^ cid ^ "\", " ^ bigSz ^ ");\n  " ^
  "clam_calc_setmatrix(" ^ ident ^ ", " ^ smallSz ^ ", " ^ (c_of_matrix mat) ^ ")"

let c_of_declare_cstring cid t str ids =
  let ident = id_of_calcId cid in
  ident ^ " = clam_calc_alloc(\"" ^ cid ^ "\", " ^ (fst (c_of_atom t)) ^ ")"

let rec c_of_calcEx = function
    CChain(ca) -> c_of_calcAssign ca
  | CIdent(id,t) -> id_of_calcId id
  | _ -> raise(Failure("Backend found an unnamed Calc expression where it shouldn't be?"))

and c_of_calcAssign ca =
  match ca.c_rhs with
    CMatrix(m) -> c_of_declare_matrix ca.c_lhs ca.c_typ m
  | CRaw(s,ids) -> c_of_declare_cstring ca.c_lhs ca.c_typ s ids
  | _ -> raise(Failure("Backend trying to assign Calc to non-matrix and non-cstring?"))

let rec c_of_kernEx = function
    KCalcList(ids,unused,assignTo) -> c_of_kernCalc assignTo unused ids
  | KChain(ka) -> c_of_kernAssign ka
  | KAppend(kap) -> c_of_kernAppend kap
  | KIdent(id) -> id_of_kernId id

and c_of_kernAssign ka =
  let kernel_needs_cloning = function
      KCalcList(_,_,_) -> false
    | KChain(_) -> true
    | KAppend(_) -> false
    | KIdent(_) -> true
  in
  let c_rhs =
    if (kernel_needs_cloning ka.k_rhs)
    then "clam_kernel_copy( (" ^ (c_of_kernEx ka.k_rhs) ^ ") )"
    else c_of_kernEx ka.k_rhs
  in
  "clam_kernel_assign(" ^ (id_of_kernId ka.k_lhs) ^ ", (" ^ c_rhs ^ ") )"

(* This appends a _used_ calculation to a Kernel object *)
and c_of_kernAppend kap =
  "TEMP_clam_kernel_addcalc(" ^ (id_of_kernId kap.ka_lhs) ^ ", (" ^ (c_of_calcEx kap.ka_rhs) ^ ") )"

let c_of_conv kid idx =
  "__convolution"^(string_of_int idx)^"("^(id_of_kernId kid)^")"

let c_of_imgread fid =
  "imgread(" ^ (c_of_fid fid) ^ ")"

let name_of_calcEx = function
    CChain(ca) -> ca.c_lhs,ca.c_typ
  | CIdent(cid,t) -> cid,t
  | _ -> raise(Failure("Backend can't find name of calc expression"))

let rec c_of_imAssign ia =
  let img_needs_cloning = function
      ImConv(_,_,_,_) -> false
    | ImRead(_)     -> false
    | ImChain(_)     -> true
    | ImAppend(_)  -> true
    | ImIdent(_)     -> true
  in
  let c_rhs =
    if (img_needs_cloning ia.i_rhs)
    then "clam_img_copy( (" ^ (c_of_imgEx ia.i_rhs) ^ ") )"
    else c_of_imgEx ia.i_rhs
  in
  "clam_img_assign(" ^ (id_of_imgId ia.i_lhs) ^ ", " ^ c_rhs ^ ")"

and c_of_imAppend iap =
  let cid,ctype = name_of_calcEx iap.ia_rhs in
  (* XXX: This is a hack and a shortcut - should be fixed later... *)
  let calcObj = Environ.calct_of_id !Semantic.scope.venv cid in
  let imgt = Environ.imgt_of_id !Semantic.scope.venv iap.ia_lhs in
  let calclst = List.map snd imgt.ichannels in
  "({ {\n"^
  "\t\tclam_img *__IMG = "^(id_of_imgId iap.ia_lhs)^";\n"^
  (inner_c_of_cfunc_calc calclst calcObj) ^
  "\n\t}; "^(id_of_imgId iap.ia_lhs)^"; })"


and c_of_imgEx = function
    ImConv(kid,_,_,idx) -> c_of_conv kid idx
  | ImRead(fid) -> c_of_imgread fid
  | ImChain(ia) -> c_of_imAssign ia
  | ImAppend(iap) -> c_of_imAppend iap
  | ImIdent(id) -> id_of_imgId id

let rec c_of_chanRefEx = function
    ChanChain(ca) -> c_of_chanAssign ca
  | ChanIdent(cid) -> id_of_chanT cid

and c_of_chanAssign ca =
  "clam_imgchan_copy( " ^
  (id_of_imgId(fst ca.ch_lhs)) ^
  ", " ^
  "\"" ^ (escaped(snd ca.ch_lhs)) ^ "\"" ^
  ", " ^
  (c_of_chanRefEx ca.ch_rhs) ^ ")"

let c_of_imgWrite ie fmt fid =
  "imgwrite( (" ^ (c_of_imgEx ie) ^ ") , " ^ (c_of_fmt fmt) ^ " , " ^ (c_of_fid fid) ^ " )"



(*
 * Glue
 *)
let c_of_scope scope =
  let venv = scope.venv in
    (List.fold_left (^) "" (List.map c_of_imgDecl venv.images)) ^
    (List.fold_left (^) "" (List.map c_of_kernDecl venv.kernels)) ^
    (List.fold_left (^) "" (List.map c_of_calcDecl venv.calc)) ^
    (List.fold_left (^) "" (List.map c_of_convData scope.cvdata))

let c_of_vExpr = function
    Debug(s) -> "/* DEBUG: " ^ s ^ " */"
  | CalcEx(ce) -> c_of_calcEx ce
  | KernelEx(ke) -> c_of_kernEx ke
  | ImageEx(ie) -> c_of_imgEx ie
  | ChanRefEx(che) -> c_of_chanRefEx che
  | ImgWriteEx(ie,fmt,fid) -> c_of_imgWrite ie fmt fid

let c_of_vStmt vExpr =
  "  " ^ (c_of_vExpr vExpr) ^ ";\n"

let generate_c scope sast =
  "\n/* CLAM: Standard C-Library Support */\n" ^
  Clam_clib.clibheader ^
  "\n/* CLAM: Generated Environment (vars/convolution funcs)*/\n" ^
  (c_of_scope scope) ^
  "\n/* CLAM: Generated main() */\n" ^
  "int main(int argc, char **argv) {\n" ^
  let m = string_of_int scope.max_arg in
  "  if (argc <= " ^ m ^ ") {\n" ^
  "    fprintf(stderr, \"This program requires " ^ m ^ " arguments.\\n\");\n" ^
  "    exit(1);\n" ^
  "  }\n" ^
  (List.fold_left (^) "" (List.map c_of_vStmt sast)) ^
  "\n  return 0;\n" ^
  "}\n"
```

```
(*
 * File: clamsys.ml
 * Date: 2011-10-17
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)
open Printf

(* assume gcc is in the path *)
let gcc_path = "g++"

(* will be something like "clamlib.o" - need to hook in auto compilation during
 * compiler build before we actually set this variable *)
let clam_extralib = "clam.a"

exception Compile_error of string

(* Execute a system command and return the output (including stderr).
 * Code shamelessly ripped from: http://rosettacode.org/wiki/Execute_a_system_command#OCaml
 *)
let syscall_check_exit_status procname stderr = function
  | Unix.WEXITED 0 -> ()
  | Unix.WEXITED r ->
      raise (Compile_error (sprintf "%s terminated with exit code (%d)\n  %s"
                                    procname r stderr))
  | Unix.WSIGNALED n ->
      raise (Compile_error (sprintf "%s was killed by a signal (%d)\n%!"
                                    procname n))
  | Unix.WSTOPPED n ->
      raise (Compile_error (sprintf "%s was stopped by a signal (%d)\n%!"
                                    procname n))
;;

let syscall ?(env=(Unix.environment ())) cmd =
  let ic, oc, ec = Unix.open_process_full cmd env in
  let buf1 = Buffer.create 96
  and buf2 = Buffer.create 48 in
  (try
     while true do Buffer.add_channel buf1 ic 1 done
   with End_of_file -> ());
  (try
     while true do Buffer.add_channel buf2 ec 1 done
   with End_of_file -> ());
  let exit_status = Unix.close_process_full (ic, oc, ec) in
  syscall_check_exit_status ("‘"^cmd^"‘") ("E:"^(Buffer.contents buf2)) exit_status;
  (Buffer.contents buf1,
   Buffer.contents buf2)
;;

(*
 * Wrap up gcc so we can output a binary!
 *)
let compile_c code oname =
  let fname, ochan = Filename.open_temp_file "clam-cc-" ".c" in
  let _ = Pervasives.output_string ochan code in
  let _ = Pervasives.close_out ochan in
  let lpath = Filename.dirname (Array.get Sys.argv 0) in
  let _, _ = syscall (sprintf "%s -c -o %s.o %s" gcc_path fname fname) in
  let _, _ = syscall (sprintf "%s -o %s -L. %s.o %s/%s"
                              gcc_path oname fname lpath
                              clam_extralib) in
  Sys.remove fname; Sys.remove (fname^".o");
  ()
```

```
(*
 * File: environ.ml
 * Date: 2011-11-09
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

open Ast
open Envtypes

let default_matrix () = (BInt(1),BInt(1)),[[BInt(1)]]

let default_image nm =
  { iname = nm;
    ichannels =
    [
      "Red",
      { cname = "Red";
        ctype = Uint8;
        cisvalid = true;
        cismat = false;
        cfunc = "",[];
        cmatrix = default_matrix (); };
      "Green",
      { cname = "Green";
        ctype = Uint8;
        cisvalid = true;
        cismat = false;
        cfunc = "",[];
        cmatrix = default_matrix (); };
      "Blue",
      { cname = "Blue";
        ctype = Uint8;
        cisvalid = true;
        cismat = false;
        cfunc = "",[];
        cmatrix = default_matrix (); };
    ];
  }

(* Add a variable definition to the environment:
 * raises a "Failure" exception if the name isn't unique
 *)
let do_add_ident env vd =
  let add_unique_var id =
        if (List.exists (fun c -> c = id) env.allvars)
        then (raise (Failure("variable re-defined: "^id)))
        else env.allvars <- id :: env.allvars
  in
  match vd with
    ImageT(nm) ->  add_unique_var nm
  | KernelT(nm) -> add_unique_var nm
  | CalcT(nm,t) -> add_unique_var nm
  | _ -> ()

(* add a strongly-typed variable (for easier/faster searching) *)
let rec var_add env vd = do_add_ident env vd;
  match vd with
    ImageT(nm) -> let rec add_unique_img = function
        [] -> [ default_image nm ]
      | hd :: tl -> if hd.iname = nm then
                      raise (Failure("ImageT redefined: "^nm))
                    else hd :: add_unique_img tl
      in
      let env1 = { env with images = add_unique_img env.images } in
      env1
  | KernelT(nm) -> let rec add_unique_kernel = function
        [] -> [ { kname = nm;
                  kallcalc = []; kunusedcalc = []; } ]
      | hd :: tl -> if hd.kname = nm then
                      raise (Failure("KernelT redefined: "^nm))
                    else hd :: add_unique_kernel tl
      in
      let env1 = { env with kernels = add_unique_kernel env.kernels } in
      env1
  | CalcT(nm,t) -> let rec add_unique_calc = function
       [] -> [ { cname = nm;
                 ctype = t;
                 cisvalid = false;
                 cismat = false;
                 cfunc = "",[];
                 cmatrix = default_matrix ();} ]
      | hd :: tl -> if hd.cname = nm then
                      raise (Failure("CalcT redefined: "
                      ^nm^"<"^(Printer.string_of_atom t)^">"))
                    else hd :: add_unique_calc tl
      in
      let env1 = { env with calc = add_unique_calc env.calc } in
      env1
  | ConvT(_,_) | KCalcT(_) | StrT(_,_) | BareT(_) -> env

let imgt_of_id env name =
  List.find (fun i -> i.iname = name) env.images

let calct_of_id env name =
  List.find (fun c -> c.cname = name) env.calc

let kernt_of_id env name =
  List.find (fun k -> k.kname = name) env.kernels

(* Find the type of the named variable:
 * raises a "Failure" exception if it's undefined
 *)
let type_of env varname =
    if (List.exists
         (fun c -> if c.cname = varname then true else false)
         env.calc)
    then
      let cval = calct_of_id env varname in CalcT(varname, cval.ctype)
    else if (List.exists
              (fun i -> if i.iname = varname then true else false)
              env.images) then ImageT(varname)
    else if (List.exists
              (fun k -> if k.kname = varname then true else false)
              env.kernels) then KernelT(varname)
    else (raise (Failure("Undefined variable: "^varname)))

(* Find the type of an expression i.e. the expression _results_ in this type *)
let rec type_of_expr env = function
    Id(i) -> type_of env i
  | Integer(BInt(i)) -> BareT("INT")
  | LitStr(s) -> StrT(":litstr", s)
  | CStr(s,idl) -> StrT(":cstr", s)
  | KernCalc(k) -> KCalcT(k)
  | ChanMat(m) -> CalcT(":c", Uint8)
  | ChanRef(c) -> CalcT(c.channel, Uint8)
  | Convolve(a,b) -> ConvT(a,b)
  | Assign(i,op,v) -> type_of_expr env v
  | ChanAssign(ref,v) -> type_of_expr env v
  | LibCall(f,args) ->
        let ctype = function
            ImgRead -> ImageT(":i")
          | ImgWrite -> BareT("VOID") in
        ctype f
```

```
(*
 * File: envtypes.mli
 * Date: 2011-11-09
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

open Ast

(* Environment types for verifier *)
type calcT = {
  cname    : string;
  ctype    : Ast.atom;
  mutable cisvalid : bool;
  mutable cismat   : bool; (* if true, use 'matrix' else use 'cfunc' *)
  mutable cfunc    : string * string list;
  mutable cmatrix  : Ast.matrix;
}

type imgT = {
  iname     : string;
  mutable ichannels : (string * calcT) list;
}

type kernelT = {
  kname       : string;
  mutable kallcalc    : string list;
  mutable kunusedcalc : string list;
}

type convRefT = {
  cvchan : string * string;
  cvkernel : kernelT;
  cvidx : int;
}

type envT = {
  mutable calc : calcT list;
  conv : convRefT list;
  images  : imgT list;
  kernels : kernelT list;
  mutable allvars : string list;
}
```

```ocaml
(*
 * File: parseutil.ml
 * Date: 2011-11-14
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

(* adapted from: http://repo.or.cz/w/sqlgg.git *)

open Scanner

exception ParseErr of exn * (string * int * int * string * string)

let parse_buf_exn lexbuf fname =
  try
    lexbuf.Lexing.lex_curr_p <- {
        Lexing.pos_fname = fname;
        Lexing.pos_lnum = 1;
        Lexing.pos_bol = 1;
        Lexing.pos_cnum = 1;
    };
    Parser.program Scanner.token lexbuf
  with exn ->
    begin
      let curr = lexbuf.Lexing.lex_curr_p in
      let file = curr.Lexing.pos_fname in
      let line = curr.Lexing.pos_lnum in
      let cnum = curr.Lexing.pos_cnum - curr.Lexing.pos_bol in
      let tok = Lexing.lexeme lexbuf in
      let tail = Scanner.tokTail "" lexbuf in
      raise (ParseErr (exn,(file,line,cnum,tok,tail)))
    end

let parse_stdin () = parse_buf_exn (Lexing.from_channel stdin) "<stdin>"

let parse_string str = parse_buf_exn (Lexing.from_string str) "<string>"

let parse_file filename =
  let chan = Pervasives.open_in filename in
  let prog = parse_buf_exn (Lexing.from_channel chan) filename in
  Pervasives.close_in chan; prog
```

```
%{(*
   * File: parser.mly
   * Date: 2011-10-10
   *
   * PLT Fall 2011
   * CLAM Project
   * Jeremy C. Andrus <jeremya@cs.columbia.edu>
   * Robert Martin <rdm2128@columbia.edu>
   * Kevin Sun <kfs2110@columbia.edu>
   * Yongxu Zhang <yz2419@columbia.edu>
   *)


open Ast %}

%token SEMI LPAREN RPAREN LTCHAR GTCHAR LBRKT RBRKT LBRACE RBRACE
%token COLON COMMA FSLASH
%token CONVOP PIPE ATSYM UMINUS UPLUS
%token ASSIGN DEFINE OREQUAL
%token IMAGET KERNELT CALCT
%token UINT8T UINT16T UINT32T INT8T INT16T INT32T ANGLET
%token IMGREAD IMGWRITE
%token <string> LITSTR
%token <string * string list> CSTR
%token <int> INTEGER
%token <string> ID
%token EOF

%right ASSIGN DEFINE OREQUAL
%left PIPE COMMA
%left CONVOP
%left COLON
%right UMINUS UPLUS ATSYM

%start program
%type <Ast.program> program

%%

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

atom:
    UINT8T  { Uint8  }
  | UINT16T { Uint16 }
  | UINT32T { Uint32 }
  | INT8T   { Int8   }
  | INT16T  { Int16  }
  | INT32T  { Int32  }
  | ANGLET  { Angle  }

libfunc:
    IMGREAD     { ImgRead }
  | IMGWRITE    { ImgWrite }

chanref:
    ID COLON ID { { image = $1; channel = $3; } }

bareint:
    INTEGER                       { BInt($1)  }
  | UMINUS INTEGER                { BInt(-$2) }
  | UPLUS INTEGER                 { if $2 < 0 then
                                      BInt(0-$2) else
                                      BInt($2) }

/* tuple:
 *   fst = list of IDs (channels) to calculate
 *   snd = list of IDs (channels) whose output is discarded
 */
kerncalc:
  | ATSYM ID               { { allcalc = [$2]; unusedcalc = [$2] } }
  | PIPE ID                { { allcalc = [$2]; unusedcalc = [] } }
  | PIPE ATSYM ID          { { allcalc = [$3]; unusedcalc = [$3] } }
  | kerncalc PIPE ID       { { allcalc = ($3 :: ($1).allcalc);
                               unusedcalc = ($1).unusedcalc } }
  | kerncalc PIPE ATSYM ID { { allcalc = ($4 :: ($1).allcalc);
                               unusedcalc = ($4 :: ($1).unusedcalc) } }

matrix_scale:
    LBRKT bareint FSLASH bareint RBRKT { ($2, $4) }

matrix_row:
    bareint { [$1] }
  | matrix_row bareint { $2 :: $1 }

matrix_start:
  | LBRACE matrix_row { [List.rev $2] }
  | matrix_start COMMA matrix_row {  (List.rev $3) :: $1 }

matrix:
  | matrix_start RBRACE { $1 }

vdecl:
    IMAGET ID                   { ImageT($2) }
  | KERNELT ID                  { KernelT($2) }
  | CALCT ID                    { CalcT($2, Uint8) }
  | CALCT ID LTCHAR atom GTCHAR { CalcT($2, $4) }

expr:
    ID                           { Id($1) }
  | bareint                      { Integer($1) }
  | LITSTR                       { LitStr($1) }
  | CSTR                         { CStr(fst $1,snd $1) }
  | kerncalc                     { KernCalc($1) }
  | matrix_scale matrix          { ChanMat($1, List.rev $2) }
  | matrix                       { ChanMat((BInt(1),BInt(1)), List.rev $1) }
  | chanref                      { ChanRef($1) }
  | chanref CONVOP ID            { Convolve($1, $3) }
  | ID ASSIGN expr               { Assign($1, Eq, $3) }
  | ID OREQUAL expr              { Assign($1, OrEq, $3) }
  | chanref ASSIGN expr          { ChanAssign($1, $3) }
  | libfunc LPAREN libfunc_args RPAREN { LibCall($1, $3) }

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

stmt:
    expr SEMI                    { Expr($1)     }
  | vdecl SEMI                   { VDecl($1)    }
  | vdecl DEFINE expr SEMI       { VAssign($1, DefEq, $3) }
  | vdecl ASSIGN expr SEMI       { VAssign($1, Eq, $3) }
```

```
(*
 * File: printer.ml
 * Date: 2011-12-02
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

open ExtString
open Ast
open Envtypes
open Sast

(* Strings that represent CLAM things *)
let string_of_op = function
    Eq -> "="
  | OrEq -> "|="
  | DefEq -> ":="

let string_of_atom = function
    Uint8 -> "U8"
  | Uint16 -> "U16"
  | Uint32 -> "U32"
  | Int8 -> "I8"
  | Int16 -> "I16"
  | Int32 -> "I32"
  | Angle -> "Angle"
  | Unknown -> "Unknown"

let string_of_libf = function
    ImgRead -> "ImgRead"
  | ImgWrite -> "ImgWrite"

let string_of_vdecl = function
    ImageT(nm) -> "Image("^nm^")"
  | KernelT(nm) -> "KernelT("^nm^")"
  | KCalcT(k) -> "KCalc"
  | CalcT(nm,t) -> "CalcT("^nm^")"
  | StrT(t, s) -> t^":"^s
  | BareT(s) -> s
  | ConvT(a,b) -> "Convolution"


let string_of_type = function
    CalcType(t) -> "Calc<" ^ (string_of_atom t) ^ ">"
  | KernelType -> "Kernel"
  | ImageType -> "Image"
  | ChanRefType -> "ChanRef"
  | FilenameType -> "Filename"
  | FormatType -> "Image Format"
  | VoidType -> "Void"

let string_of_chan ch =
  ch.image ^ "." ^ ch.channel

(* Printing CLAM compiler messages *)
let print_clamerr = function
    Parseutil.ParseErr(exn,(file,line,cnum,tok,tail)) ->
      let extra = Printexc.to_string exn in
      let fname = if file = "" then "<stdin>" else file in
      let estr =
        if tok = "" then
          Printf.sprintf "%s" extra
        else
          Printf.sprintf "%s at %s:%u:%u near \"%s%s\""
            extra fname line cnum tok (String.slice ~last:32 tail)
        in
      prerr_endline estr;
  | _ -> ()

(* Environment Printing *)
let string_of_scope scope =
  let _ = scope.venv in
    "Environment:\n" ^
    "  Printer not implemented\n"

```

```
let print_scope scope =
  print_endline (string_of_scope scope)

(* CLAM AST Printing *)
type 'a ptree = Node of 'a * ('a ptree list)

let tree_of_atom a =
  Node("Atomic Type '" ^ (string_of_atom a) ^ "'", [])

let tree_of_assign_op op =
  Node("Assignment Op '" ^ (string_of_op op) ^ "'", [])

let tree_of_bareint bi =
  match bi with
    BInt(i) -> Node("BareInt: " ^ string_of_int(i), [])

let tree_of_ident id =
  Node("Identifer '" ^ id ^ "'", [])

let tree_of_kerncalc kc =
  Node("? KernCalc", [])

let tree_of_chanref ref =
  Node("? ChanRef", [])

let tree_of_chanmat mat =
  Node("? ChanMat", [])

let tree_of_libf libf =
  Node("Library Function: " ^ (string_of_libf libf), [])

let tree_of_vdecl vdecl =
  match vdecl with
      ImageT(id) -> Node("Variable Declaration [Image Type]", [tree_of_ident id])
    | KernelT(id) -> Node("Variable Declaration [Kernel Type]", [tree_of_ident id])
    | CalcT(id, a) -> Node("Variable Declaration [Calc Type]", [tree_of_ident id; tree_of_atom a])
    | KCalcT(k) -> Node("INVALID use of KCalcT", [])
    | _ -> Node("Invalid use of ...", [])

let rec tree_of_expr expr =
  let tupl = match expr with
      Id(id) -> ("Identifier", [tree_of_ident id])
    | Integer(bi) -> ("Integer", [tree_of_bareint bi])
    | LitStr(s) -> ("LitStr", [Node("'" ^ s ^ "'", [])])
    | CStr(s,idl) -> ("CStr", [Node("'" ^ s ^ "'", [])])
    | KernCalc(kc) -> ("KernCalc", [tree_of_kerncalc kc])
    | ChanMat(mat) -> ("ChanMat", [tree_of_chanmat mat])
    | ChanRef(ref) -> ("ChanRef", [tree_of_chanref ref])
    | Convolve(ref, kid) ->
        ("Convolve", [tree_of_chanref ref; tree_of_ident kid])
    | Assign(id, op, e) ->
        ("Assign", [tree_of_ident id; tree_of_assign_op op; tree_of_expr e])
    | ChanAssign(ref, e) ->
        ("ChanAssign", [tree_of_chanref ref; tree_of_expr e])
    | LibCall(libf, elist) ->
        ("LibCall", List.append [tree_of_libf libf] (List.map tree_of_expr elist))
  in
  Node("Expression [" ^ fst(tupl) ^ "]", snd(tupl))

let tree_of_stmt stmt =
  let children = match stmt with
      Expr(e) -> [tree_of_expr e]
    | VDecl(v) -> [tree_of_vdecl v]
    | VAssign(v, op, e) -> [tree_of_vdecl v; tree_of_assign_op op; tree_of_expr e]
  in
  Node("Statement", children)

let tree_of_ast ast =
  let children = List.map tree_of_stmt ast in
  Node("Abstract Syntax Tree", children)

let rec string_of_tree prefix = function Node(str, ch) ->
  let string_of_children =
    List.fold_left (^) "" (List.map (string_of_tree (prefix ^ "    ")) ch)
  in
  prefix ^ " * " ^ str ^ "\n" ^ string_of_children

let string_of_ast ast =
  string_of_tree "" (tree_of_ast ast)
```

```
(*
 * File: sast.ml
 * Date: 2011-10-16
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu> * Yongxu Zhang <yz2419@columbia.edu>
 *)

(* Identifiers *)
type calcId = string
type kernId = string
type imgId = string
type chanRefId = imgId * calcId
type convData = kernId * chanRefId * Envtypes.calcT list * int
type filenameId = Const of string | Arg of int

(* Environment Objects *)
type sMatrix = (int * int ) * (int * int) * int list list
type fmtType = Png | Bmp | Tga
type typeT = CalcType of Ast.atom
           | KernelType
           | ImageType
           | ChanRefType
           | FilenameType
           | FormatType
           | VoidType
type identT = {
  id: string;
  typ: typeT;
  init: bool;
  chans: string list; (* Only relevant for image identifiers *)
}

type scopeT = {
  venv: Envtypes.envT;
  mutable mats: sMatrix list;
  mutable max_arg: int;
  mutable cvdata : convData list;
}


(* Assignment to a Calc *)
type calcAssign = { c_lhs: calcId; c_rhs: calcEx; c_typ: Ast.atom; }
and calcEx =
    CMatrix of sMatrix
  | CRaw of string * calcId list
  | CChain of calcAssign
  | CIdent of calcId * Ast.atom

(* Assignment to a Kernel *)
type kernAppend = { ka_lhs: kernId; ka_rhs: calcEx; }
type kernAssign = { k_lhs: kernId; k_rhs: kernEx; }
and kernEx =
    KCalcList of calcId list * calcId list * string (* all, unused, assignTo *)
  | KChain of kernAssign
  | KAppend of kernAppend
  | KIdent of kernId

(* Assignment to an Image *)
type imgAppend = { ia_lhs: imgId; ia_rhs: calcEx; }
type imgAssign = { i_lhs: imgId; i_rhs: imgEx; }
and imgEx =
    ImConv of convData
  | ImRead of filenameId
  | ImChain of imgAssign
  | ImAppend of imgAppend
  | ImIdent of imgId

(* Assignment to a Channel Reference *)
type chanAssign = { ch_lhs: chanRefId; ch_rhs: chanRefEx; }
and chanRefEx =
    ChanChain of chanAssign
  | ChanIdent of chanRefId

(* Output images *)
type imgWrite = { im: imgEx; fil: filenameId; fmtType: fmtType; }

type vExpr =
    CalcEx of calcEx
  | KernelEx of kernEx
  | ImageEx of imgEx
  | ChanRefEx of chanRefEx
  | ImgWriteEx of imgEx * fmtType * filenameId
  | Debug of string

type vastRoot = vExpr list
```

```
(*
 * File: scanner.mll
 * Date: 2011-10-11
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)
{ open Parser

exception LexError of string

(* string parsing from OCaml compiler code :-) *)
let string_buff = Buffer.create 256
let reset_string_buffer () = Buffer.clear string_buff
let store_string_char c = Buffer.add_char string_buff c
let store_string_snip str = Buffer.add_string string_buff str
let get_stored_string () = Buffer.contents string_buff

(* ID list for 'escaped C string' parsing *)
let idlist = ref []
let reset_idlist () = idlist := []
let add_id_to_list id =
  if not (List.exists (fun i -> if i = id then true else false) !idlist) then
    idlist := id :: !idlist
  else ()

let char_for_backslash = function
    'n' -> '\n'
  | 't' -> '\t'
  | 'b' -> '\b'
  | 'r' -> '\r'
  | c   -> c

let decimal_code c d u =
  100 * (Char.code c - 48) + 10 * (Char.code d - 48) + (Char.code u - 48)

let char_for_hexadecimal_code d u =
  let d1 = Char.code d in
  let val1 = if d1 >= 97 then d1 - 87
             else if d1 >= 65 then d1 - 55
             else d1 - 48
  in
  let d2 = Char.code u in
  let val2 = if d2 >= 97 then d2 - 87
             else if d2 >= 65 then d2 - 55
             else d2 - 48
  in
  Char.chr (val1 * 16 + val2)

let lex_warning lexbuf msg =
  let p = Lexing.lexeme_start_p lexbuf in
  Printf.eprintf "CLAM warning:\nFile \"%s\", line %d, character %d: %s.\n"
    p.Lexing.pos_fname p.Lexing.pos_lnum
    (p.Lexing.pos_cnum - p.Lexing.pos_bol + 1) msg;
  flush stderr

let incr_loc lexbuf delta =
  let pos = lexbuf.Lexing.lex_curr_p in
  lexbuf.Lexing.lex_curr_p <- { pos with
    Lexing.pos_lnum = pos.Lexing.pos_lnum + 1;
    Lexing.pos_bol = pos.Lexing.pos_cnum - delta;
  }
;;
}

let newline    = '\n' | "\r\n"
let whitespace = [' ' '\t']
let digit      = ['0'-'9']
let integer    = digit+
let alpha      = ['_' 'a'-'z' 'A'-'Z']
let alphanum   = alpha | digit
let identifier = alpha alphanum*
let backslash_escapes = ['\\' '"' '\'' 'n' 't' 'b' 'r']
let invalidcstr_char  = ['{' '}' ';' '#' '"' ''' ] | "/*" | "*/" | "//"
let cstr_cast = ['('] whitespace* alpha alphanum* whitespace* [')']
let cstr_libcall = alpha alphanum* ['(']
(* mash consecutive strings together right in the scanner! *)
let consecutive_strings = ['"'] whitespace* ['"']

rule token = parse
    newline             { Lexing.new_line lexbuf; token lexbuf }
  | whitespace          { token lexbuf }
  | "/*"                { comment 0 lexbuf }

  (* string parsing from OCaml compiler code :-) *)
  | '"'
    { reset_string_buffer ();
      parse_string lexbuf;
      (*handle_lexical_error string lexbuf;*)
      LITSTR(get_stored_string ()) }

  | "#["
    { reset_string_buffer ();
      reset_idlist ();
      parse_cstr lexbuf;
      CSTR(get_stored_string (), !idlist) }

  (* operators *)
  | "**"                { CONVOP   }
  | ":="                { DEFINE   }
  | "|="                { OREQUAL  }
  | '='                 { ASSIGN   }
  | '|'                 { PIPE     }
  | '@'                 { ATSYM    }
  | '-'                 { UMINUS   }
  | '/'                 { FSLASH   }
  | '+'                 { UPLUS    }

  (* punctuation *)
  | '('                 { LPAREN   }
  | ')'                 { RPAREN   }
  | '<'                 { LTCHAR   }
  | '>'                 { GTCHAR   }
  | '['                 { LBRKT    }
  | ']'                 { RBRKT    }
  | '{'                 { LBRACE   }
  | '}'                 { RBRACE   }
  | ';'                 { SEMI     }
  | ':'                 { COLON    }
  | ','                 { COMMA    }

  (* built-in types *)
  | "Image"             { IMAGET   }
  | "Kernel"            { KERNELT  }
  | "Calc"              { CALCT    }
  | "Uint8"             { UINT8T   }
  | "Uint16"            { UINT16T  }
  | "Uint32"            { UINT32T  }
  | "Int8"              { INT8T    }
  | "Int16"             { INT16T   }
  | "Int32"             { INT32T   }
  | "Angle"             { ANGLET   }

  (* library functions *)
  | "imgread"           { IMGREAD  }
  | "imgwrite"          { IMGWRITE }

  | integer as lit      { INTEGER(int_of_string lit) }
  | identifier as lit   { ID(lit) }
  | eof                 { EOF }
  | _ as char           { raise (LexError("illegal character '" ^
                                          (Char.escaped char^"'"))) }

(* fancy string parsing from OCaml compiler code :-) *)
and parse_string = parse
    consecutive_strings  { parse_string lexbuf }
  | '"'     { () }
  | newline { Lexing.new_line lexbuf; parse_string lexbuf }
  | '\\' ("\010" | "\013" | "\013\010") ([' ' '\009'] * as spaces)
    { incr_loc lexbuf (String.length spaces);
      parse_string lexbuf }
  | '\\' (backslash_escapes as c)
    { store_string_char(char_for_backslash c);
      parse_string lexbuf }
  | '\\' 'x' (['0'-'9' 'a'-'f' 'A'-'F'] as d) (['0'-'9' 'a'-'f' 'A'-'F'] as u)
    { store_string_char (char_for_hexadecimal_code d u) ;
      parse_string lexbuf }
  | '\\' (['0'-'9'] as c) (['0'-'9'] as d) (['0'-'9']  as u)
    { let v = decimal_code c d u in
      if v > 255 then
       lex_warning lexbuf
        (Printf.sprintf
          "illegal backslash escape in string: `\\%c%c%c'" c d u) ;
      store_string_char (Char.chr v);
      parse_string lexbuf }
  | '\\' (_ as c)
    { lex_warning lexbuf
        (Printf.sprintf "illegal backslash escape in string: `\\%c'" c) ;
      store_string_char '\\' ;
      store_string_char c ;
      parse_string lexbuf }
  | '\010'
    { store_string_char '\010';
      incr_loc lexbuf 0;
      parse_string lexbuf }
  | eof { raise(LexError("unterminated string")) }
  | _ as c
    { store_string_char c;
      parse_string lexbuf }

(* handle escaped-C sequences: do some basic sanity parsing
 * (to ensure that nothing crazy is going on) *)
and parse_cstr = parse
    "]#"    { () }
  | newline            { Lexing.new_line lexbuf; parse_cstr lexbuf }
  | whitespace         { parse_cstr lexbuf }
  | ")"
        { raise (LexError("Unmatched ')' in escaped C")) }
  | invalidcstr_char
        { raise (LexError("Invalid character in escaped-C string")) }
  | cstr_cast as cast
        { store_string_snip cast; parse_cstr lexbuf }
  | cstr_libcall | "(" as str
        { store_string_snip str; parse_cstr_libcall 0 lexbuf }
  | identifier as id
        { store_string_snip id; add_id_to_list id; parse_cstr lexbuf }
  | eof
        { raise (LexError("unterminated escaped-C string!")) }
  | _ as c
        { store_string_char c;
          parse_cstr lexbuf }

and parse_cstr_libcall level = parse
    "]#" { raise (LexError("Mismatched parens in escaped-C string")) }
  | newline            { Lexing.new_line lexbuf; parse_cstr_libcall level lexbuf }
  | whitespace         { parse_cstr_libcall level lexbuf }
  | ")" { store_string_char ')';
          if level = 0 then
            parse_cstr lexbuf
          else
            parse_cstr_libcall (level-1) lexbuf }
  | invalidcstr_char
        { raise (LexError("Invalid character in escaped-C string")) }
  | cstr_cast as cast
        { store_string_snip cast; parse_cstr_libcall (level) lexbuf }
  | cstr_libcall | "(" as str
        { store_string_snip str;
          parse_cstr_libcall (level+1) lexbuf }
  | identifier as id
        { store_string_snip id; add_id_to_list id; parse_cstr_libcall level lexbuf }
  | eof { raise (LexError("unterminated function call in escaped-C string")) }
  | _ as c
    { store_string_char c;
      parse_cstr_libcall level lexbuf }

and comment level = parse
    "*/"  { if level = 0 then token lexbuf
            else comment (level-1) lexbuf }
  | newline { Lexing.new_line lexbuf; comment level lexbuf }
  | "/*"    { comment (level+1) lexbuf }
  | eof     { raise (LexError("unterminated comment!")) }
  | _       { comment level lexbuf }

and tokTail acc = parse
    eof { acc }
  | _* as str { tokTail (acc ^ str) lexbuf }
```

```
(*
 * File: sematic.ml
 * Date: 2011-12-08
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

open Ast
open Sast
open Envtypes
open Environ
open Printer

(* Strategy:
 * trans_N ENV N =
 *   1) Make sure N has the correct child nodes
 *   2) Check the child nodes
 *   3) Update the ENV for any effects from N
 *   4) Create VAST, the vast node that represents N
 *   5) Return (ENV, VAST)
 *)
exception SemanticFailure of string

let globalConvIdx = ref 0

let scope = ref {
              mats = [];
              venv = { calc = []; images = []; kernels = []; conv = []; allvars = []; };
              max_arg = 0;
              cvdata = [];
            }

let type_of_vdecl = function
    ImageT(s) -> ImageType
  | KernelT(s) -> KernelType
  | KCalcT(kc) ->
      raise(SemanticFailure("Kernel Calc does not have an associated type. "^
                            "Does this VDecl even exist!?"))
  | ConvT(e1,e2)-> ImageType
  | CalcT(s,t) -> CalcType(t)
  | _ -> raise(SemanticFailure("Invalid use of internal type."))

let type_of_vexpr = function
    (* XXX: We have problems in how we handle the atomic types of Calcs *)
    CalcEx(e) -> CalcType(Unknown)
  | KernelEx(e) -> KernelType
  | ImageEx(e) -> ImageType
  | ChanRefEx(e) -> ChanRefType
  | ImgWriteEx(im,f,fi) -> VoidType
  | Debug(s) ->
      print_endline("XXX: pretending Debug is a type CalcType(Unknown)"); CalcType(Unknown)


let ident_of_vdecl = function
    ImageT(s) -> s
  | CalcT(s,t) -> s
  | KernelT(s) -> s
  | KCalcT(kc) -> raise(SemanticFailure("Kernel Calc does not have an associated identifier string"))
  | ConvT(e1,e2)-> raise(SemanticFailure("Convolution does not have an associated identifier string"))
  | _ -> raise(SemanticFailure("Invalid use of internal type!"))

let type_of_ident scope s =
  type_of_vdecl (Environ.type_of !scope.venv s)

let int_of_BInt = function
  BInt(i) -> i

let check_max_arg i =
  if (!scope.max_arg < i) then (!scope.max_arg <- i; ())
  else ()

(*
 * Recursive Checking Functions
 *)

```

```
(* Returns: filenameId *)
let filenameId_of_expr = function
    Integer(bi) -> let i = int_of_BInt(bi) in check_max_arg i; Arg(i)
  | LitStr(s) -> Const(s)
  | _ -> raise(SemanticFailure("Filenames can only be a string or an integer"))

(* Returns fmtType *)
let fmtType_of_expr = function
    LitStr(s) -> (match s with
        "png" -> Png
      | "bmp" -> Bmp
      | "tga" -> Tga
      | _ -> raise(SemanticFailure("Unknown image format: " ^ s))
    )
  | _ -> raise(SemanticFailure("Image format must be specified as a string literal"))

(* Returns: vExpr *)
let trans_id s =
  let typ = type_of_ident scope s in
    match typ with
        CalcType(t) -> CalcEx(CIdent(s, t))
      | KernelType  -> KernelEx(KIdent(s))
      | ImageType   -> ImageEx(ImIdent(s))
      | _ -> raise(SemanticFailure("Environment claimed identifier was non-standard type"))

(* Returns: CMatrix *)
let cMatrix_of_matrix m =
  let ((bNum, bDen), bColRow) = m in
    let num = int_of_BInt bNum in
    let den = int_of_BInt bDen in
    let height = List.length bColRow in
    let width = List.length (List.hd bColRow) in
    let colRow = List.map (List.map int_of_BInt) bColRow in
      CMatrix((width, height), (num, den), colRow)



(* Returns: chanRefId *)
let trans_chanRefIdLval ch = ( ch.image , ch.channel )

(* Returns: chanRefId *)
let trans_chanRefId ch = ( ch.image , ch.channel )

(* Returns: vExpr *)
let trans_imgread elist = ImageEx(ImRead(filenameId_of_expr (List.hd elist)))

(* Returns: vExpr *)
let rec imgwrite_of_elist elist =
  match elist with
      fname_expr :: fmt_expr :: img_expr :: [] -> ( (* Yes. They're in the list backwards. *)
        let imgEx =
          let ve = trans_expr img_expr in
            match ve with   ImageEx(ie) -> ie
                          | _ -> raise(SemanticFailure("ImgWrite not passed an image?"))
        in
        let fmtType = fmtType_of_expr fmt_expr in
        let filenameId = filenameId_of_expr fname_expr in
        ImgWriteEx(imgEx, fmtType, filenameId)
      )
    | _ -> raise(SemanticFailure("Wrong number of arguments supplied to imgwrite function"))

(* Returns: vExpr *)
and trans_libf libf elist =
  match libf with
     ImgRead -> trans_imgread elist
   | ImgWrite -> (imgwrite_of_elist elist)

(* Returns: ImConv *)
and trans_conv cref id =
  let chanIdent = trans_chanRefId cref in
  let convref = (List.find (fun cv -> if (cv.cvidx = !globalConvIdx) then true else false)
                            !scope.venv.conv) in
  let chanlist = List.map (Environ.calct_of_id !scope.venv) convref.cvkernel.kallcalc in
  let imconv_stuff = id,chanIdent,chanlist,!globalConvIdx in
  let retval = ImConv(imconv_stuff) in
  globalConvIdx := !globalConvIdx + 1;
  !scope.cvdata <- imconv_stuff :: !scope.cvdata;
  retval

(* Returns: vExpr *)
and trans_expr = function
```

```
    Id(s) -> trans_id s
  | Integer(bi) -> Debug("Ignoring integer expression: " ^ (string_of_int (int_of_BInt bi)))
  | LitStr(s) -> Debug("Ignoring String literal: " ^ s)
  | CStr(s,ids) -> CalcEx(CRaw(s, ids))
  | KernCalc(kc) -> KernelEx(KCalcList(kc.allcalc,kc.unusedcalc,""))
  | ChanMat(m) -> CalcEx(cMatrix_of_matrix m)
  | ChanRef(ch) -> ChanRefEx(ChanIdent(trans_chanRefId ch))
  | Convolve(e1,e2) -> ImageEx(trans_conv e1 e2)
  | Assign(s,op,e) -> trans_assign s op e
  | ChanAssign(ch, e) -> (let chId = trans_chanRefIdLval ch in
                          let ve = trans_expr e in
                            match ve with
                                ChanRefEx(cve) ->
                                  ChanRefEx(ChanChain({ch_lhs = chId; ch_rhs = cve;}))
                              | _ -> raise(SemanticFailure("Must assign Channel to a channel type"))
                         )
  | LibCall(libf, elist) -> trans_libf libf elist

(* Returns: vExpr *)
and trans_eq_assign s e =
  let ve = trans_expr e in
    match (type_of_ident scope s) with
        CalcType(t) -> (match ve with
              CalcEx(ce) -> CalcEx(CChain({ c_lhs = s; c_rhs = ce; c_typ = t; }))
            | _ -> raise(SemanticFailure("Bad assignment")))
      | KernelType -> (match ve with
              KernelEx(ke) -> KernelEx(KChain({ k_lhs = s; k_rhs = ke; }))
            | CalcEx(t) ->
                (match t with   CIdent(cnm,typ) ->
                                  KernelEx(KChain({ k_lhs = s; k_rhs = KCalcList([cnm],[],"")}))
                              | _ -> raise(SemanticFailure("Bad Kernel Assignment"))
                )
            | _ -> raise(SemanticFailure("Bad assignment")))
      | ImageType -> (match ve with ImageEx(ie) -> ImageEx(ImChain({ i_lhs = s; i_rhs = ie; }))
                                  | _ -> raise(SemanticFailure("Bad assignment")))
      | _ -> raise(SemanticFailure("Identifier claims to be an impossible data type"))

and trans_or_assign s e =
  let ve = trans_expr e in
    match ve with
        CalcEx(c) -> (match (type_of_ident scope s) with
                  KernelType -> KernelEx(KAppend({ ka_lhs = s; ka_rhs = c; }))
                | ImageType -> ImageEx(ImAppend({ ia_lhs = s; ia_rhs = c; }))
                | _ -> raise(SemanticFailure("|= Calc must have Kernel or Image as its L-Value"))
              )
      | KernelEx(ke) -> (match (type_of_ident scope s) with
                | KernelType -> (match ke with   KCalcList(all,unused,_) ->
                                                    KernelEx(KCalcList((List.rev all),unused,s))
                                               | KIdent(ki) ->
                                                    let v_kc = kernt_of_id !scope.venv ki in
                                                    KernelEx(KCalcList((List.rev v_kc.kallcalc),
                                                                       v_kc.kunusedcalc,s))
                                               | _ -> raise (Failure("|= Kernel can't be complex?!")))
                | _ -> raise(SemanticFailure("|= Kernel must have Kernel as its L-Value")) )
      | _ -> raise(SemanticFailure("Unexpected expression is an R-Value for OrEq operation"))


(* Returns: vExpr *)
and trans_def_assign s t e = match (trans_expr e) with
    CalcEx(cexp) -> CalcEx(CChain({ c_lhs = s; c_rhs = cexp; c_typ = t; }))
  | _ -> raise(SemanticFailure("DefEq to something not a Calc expression"))

(* Returns: vExpr *)
and trans_assign s op e =
  match op with
      Eq -> trans_eq_assign s e
    | OrEq -> trans_or_assign s e
    | _ -> raise(SemanticFailure("A DefEq in an unexpected place?"))

let trans_vdecl = function
    ImageT(s)  -> Debug("Declare Image")
  | KernelT(s) -> Debug("Declare Kernel")
  | CalcT(s,t) -> Debug("Declare Calc")
  | _ -> raise(SemanticFailure("A variable declaration did not have a recognizable type"))

(* Returns: vExpr *)
let trans_action_expr expr = match expr with
    Assign(s,op,e) -> trans_expr expr
  | ChanAssign(chref,e) -> trans_expr expr
  | LibCall(libf,elist) -> trans_expr expr
  | _ -> Debug("Expression result ignored!") (* TODO: Can side effects be hiding in the expr? *)
```

```

let trans_stmt = function
    Expr(e) -> trans_action_expr e
  | VDecl(v) -> trans_vdecl v
  | VAssign(v,op,e) -> (
      let _ = trans_vdecl v in
        match v with
            CalcT(s,t) -> trans_def_assign s t e
          | _ -> trans_assign (ident_of_vdecl v) op e
    )

let translate_ast env ast =
  scope.contents <- { venv = env; mats = []; max_arg = 0; cvdata = []; };
  let gather nodes stmt = (trans_stmt stmt) :: nodes in
    let nodelist = List.fold_left gather [] ast in
      (!scope, List.rev nodelist)
```

```ocaml
(*
 * File: verifier.ml
 * Date: 2011-10-17
 *
 * PLT Fall 2011
 * CLAM Project
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * Robert Martin <rdm2128@columbia.edu>
 * Kevin Sun <kfs2110@columbia.edu>
 * Yongxu Zhang <yz2419@columbia.edu>
 *)

open Ast
open Envtypes
open Environ
open Printer

let globalConvIdx = ref 0

(*
 * Find an image by name from the environment
 *)
let find_image env imgnm = List.find
  (fun i -> if i.iname = imgnm then true else false)
  env.images

let find_kernel env knm =
  let kern = (List.find
                (fun k -> if k.kname = knm then true else false)
                env.kernels) in
  kern.kallcalc, kern.kunusedcalc

(*
 * Check if an image has a channel with the specified name
 *)
let image_has imgnm channel env simple =
  let eImg = (List.find
                (fun i -> if i.iname = imgnm then true else false)
                env.images)
  in
    if (List.exists
         (fun (nm,calc) -> if nm = channel then true else false)
         eImg.ichannels)
    then ()
    else
      if simple then
        (raise (Failure("NOCHAN")))
      else
        (raise (Failure("No channel '"^channel^"' in image '"^imgnm^"'")))

let get_calctype env chname =
  match chname with
    "Red" | "Green" | "Blue" -> Ast.Uint8
  | _ -> let c = (List.find (fun i -> if i.cname = chname then true else false)
                     env.calc)
         in c.ctype

let kcalc_add env kname allC unusedC =
  let kT = (List.find
             (fun k -> if k.kname = kname then true else false)
             env.kernels)
  in
    kT.kallcalc <- List.append kT.kallcalc allC;
    kT.kunusedcalc <- List.append kT.kunusedcalc unusedC;
    env

(* Set a matrix value in a CalcT *)
let calc_set_matrix env nm mat =
  let cval = (List.find
               (fun c -> if c.cname = nm then true else false)
               env.calc)
  in
    cval.cismat <- true;
    cval.cmatrix <- mat;
    env

(* Set the CStr in a CalcT *)
let calc_set_cfunc env nm func =
  let cval = (List.find
               (fun c -> if c.cname = nm then true else false)
               env.calc)
  in
    cval.cisvalid <- true;
    cval.cismat <- false;
    cval.cfunc <- func;
    env

let calc_ismat env nm =
  let cval = (List.find
               (fun c -> if c.cname = nm then true else false)
               env.calc)
  in cval.cismat

let calc_isvalid env nm =
  let cval = (List.find
               (fun c -> if c.cname = nm then true else false)
               env.calc)
  in cval.cisvalid

(*
 * Add a channel to the specified image
 * (the caller needs to initialize most of the member variables)
 *)
let image_add env img channel typ isvalid ismat cfuncstr cmat =
  (* add the channel, to the image *)
  img.ichannels <- List.append img.ichannels
                   [ (channel,
                     { cname = channel;
                       ctype = typ;
                       cisvalid = isvalid;
                       cismat = ismat;
                       cfunc = cfuncstr;
                       cmatrix = cmat;
                     }) ];
  env

(*
 * VERIFY:
 *   a convolution statement
 *   i.e. it checks validity of "Channel ** Kernel" statements
 *        and retuns a list of new channels produced by the
 *        convolution (so we can add them to the resulting image)
 *)
let check_convolve env chanref kref =
  let allC, unusedC = find_kernel env kref in
  let calc_is_used chname =
        if (List.exists
            (fun nm -> if nm = chname then true else false)
            unusedC)
        then false else true in
  let chanlist = (List.fold_left
                  (fun lst calc ->
                          if (calc_is_used calc) then calc :: lst else lst)
                  [] allC) in
  let cvref = { cvchan = chanref.image, chanref.channel;
                cvkernel = { kname = kref; kallcalc = allC; kunusedcalc = unusedC; };
                cvidx = !globalConvIdx;
              } in
  let env1 = { env with conv = cvref :: env.conv } in
  globalConvIdx := !globalConvIdx + 1;
  env1, chanlist

(*
 * VERIFY:
 *   our basic type checker!
 *   checks for a valid assignment from rhs to lhs
 *   i.e. it checks the expression:
 *              lhs = rhs
 *        for validity
 *)
let check_assignment env rhs rhse op = function (* passes in LHS *)
    ImageT(nm) ->
        let optype rhs = function
            DefEq -> (raise (Failure("Cannot define "^
                                     (string_of_vdecl (ImageT(nm)))^
                                     " with ':='")))
          | Eq ->
                let chk_img_assign = function
                    ConvT(a,b) ->
                        let env1, chanlst = check_convolve env a b in
                        let img = find_image env1 nm in
```

```
                      let envNew = List.fold_left
                          (fun e c -> image_add e img c (get_calctype e c)
                                     false false ("",[])
                                     ((BInt(1),BInt(1)),[[BInt(1)]]) )
                          env1 chanlst in
                      envNew
                | ImageT(nm) -> env
                | _ as t -> (raise (Failure("Can't assign '"^
                                       (string_of_vdecl t)^
                                       "' to "^(string_of_vdecl (ImageT(nm)))^
                                       ": Image = Image; only!")))
              in
              let env1 = chk_img_assign rhs in
              env1
        | OrEq ->
              let chk_chan_add = function
                  CalcT(cnm,t) ->
                      let isvalid = calc_isvalid env cnm in
                      if not isvalid then
                        (raise (Failure("Can't assign an un-initialized "^
                                        "calculation '"^
                                        cnm^ "' as an image channel")))
                      else
                        let ismat = calc_ismat env cnm in
                        if ismat then
                          (raise (Failure("Can't assign a matrix "^
                                          "calculation '"^
                                          cnm^"' to an image channel")))
                        else cnm, t
                | _ as t -> (raise (Failure("Can't assign "^
                             (string_of_vdecl t)^" to "^
                             (string_of_vdecl (ImageT(nm)))^
                             ": Invalid image channel in |=")))
              in
              let chname, chtype = chk_chan_add rhs in
              let env1 = image_add env (find_image env nm) chname chtype
                                 false false ("",[])
                                 ((BInt(1),BInt(1)),[[BInt(1)]]) in
              env1
      in optype rhs op
| KernelT(nm) ->
      let kcalc = function
            KCalcT(k) -> k
          | _ -> (raise (Failure("Internal Error."))) in
      let optype rhs = function
          DefEq -> (raise (Failure("Cannot define "^
                                    (string_of_vdecl (KernelT(nm)))^
                                   " with ':='")))
        | Eq -> ( match rhse with
                  | KernCalc(k) -> let env1 = kcalc_add env nm (List.rev k.allcalc) k.unusedcalc in
                                 env1
                  | _ -> if not (rhs = KCalcT(kcalc rhs))
                         then (raise (Failure("Can't assign "^(string_of_vdecl rhs)^
                                       " to "^(string_of_vdecl (KernelT(nm)))^
                                       ": Kernel = Kernel only!")))
                          else
                            let kc = kcalc rhs in
                            let env1 = kcalc_add env nm (List.rev kc.allcalc) kc.unusedcalc in
                            env1
                 )
        | OrEq -> let chk_calc_add = function
                        CalcT(cnm,t) -> [cnm], []
                      | KCalcT(k) -> k.allcalc, k.unusedcalc
                      | KernelT(nm) -> (try
                                          let kc = kernt_of_id env nm in
                                          kc.kallcalc, kc.kunusedcalc
                                        with _ -> raise (Failure("Undefined Kernel "^nm)) )
                      | _ as t -> (raise (Failure("Can't add "^
                                           (string_of_vdecl t)^" to "^
                                           (string_of_vdecl (KernelT(nm)))^
                                           ": Invalid CalcT!")))
                  in
                  let allC, unusedC = chk_calc_add rhs in
                  let env1 = kcalc_add env nm allC unusedC in
                  env1
      in optype rhs op
| CalcT(nm,t) ->
      let optype rhs = function
          DefEq ->
              let update_calc = function
                | ChanMat(m) -> calc_set_matrix env nm m
```

```
                   | CStr(s,idl) -> calc_set_cfunc env nm (s,idl)
                   | _ as e -> (raise (Failure("Cannot define "^
                                            (string_of_vdecl (CalcT(nm,t)))^
                                            " with '"^
                                            (string_of_vdecl (type_of_expr env e))^"'")))
                 in
                 let env1 = update_calc rhse in
                 env1
         | _ as op -> (raise (Failure("Cannot define "^
                                    (string_of_vdecl (CalcT(nm,t)))^
                                     " with '"^(string_of_op op)^"'")))
       in optype rhs op
  | ConvT(_,_) | KCalcT(_) ->
       (raise (Failure("Can't assign to internal type!")))
  | StrT(t,s) -> (raise (Failure("Can't assign to a string: '"^s^"'")))
  | BareT(s) -> (raise (Failure("Cannot assign to '"^s^"'")))

(*
 * VERIFY:
 *   a channel reference needs to reference a valid image, and must
 *   already be a defined channel on that image
 *)
let check_chanref env cref simple =
  if not (type_of env cref.image = ImageT(cref.image)) then
    if simple then
      (raise (Failure("NONIMG")))
    else
      (raise (Failure("Channel reference ("^cref.channel^
                      ") on non-image: "^cref.image)))
  else
    image_has cref.image cref.channel env simple

(*
 * VERIFY:
 *   our expression checker
 *)
let rec check_expr env = function
    Id(i) -> let _ = type_of env i in env, Id(i)
  | Integer(BInt(i)) -> env, Integer(BInt(i))
  | LitStr(s) -> env, LitStr(s)
  | CStr(s,idl) -> env, CStr(s,idl)
  | KernCalc(k) -> env, KernCalc(k)
  | ChanMat(m) -> let denom = (snd (fst m)) in
                  let matrix = snd m in
                  if (denom <> BInt(0)) then
                     let eqrows = List.fold_left
                                  (fun cols lst -> if List.length lst = cols
                                     then List.length lst else -1)
                                  (List.length (List.hd matrix)) matrix
                     in
                     if eqrows = -1 then raise (Failure("Unequal matrix rows"))
                     else env, ChanMat(m)
                  else raise(Failure("Divide by zero"))
  | ChanRef(c) -> check_chanref env c false; env, ChanRef(c)
  | Convolve(a,b) ->
       let env1, va = check_chanref env a false; env, a in
       let env2, vb = if (type_of env1 b = KernelT(b)) then env1, b else
                      (raise (Failure("Can't convolve with non-kernel "^b)))
       in
       env2, Convolve(va,vb)
  | Assign(i,op,v) ->
       let lhs = type_of env i in
       let env1, vexpr = check_expr env v in
       let rhs = type_of_expr env1 vexpr in
       let env2 = check_assignment env1 rhs vexpr op lhs in
       env2, Assign(i,op,vexpr)
  | ChanAssign(ref,v) ->
       let env1, ve = check_expr env v in
       let envNew =
         (try check_chanref env1 ref true; env1
          with Failure(s) ->
            if s = "NOCHAN" then
              let get_channame = function
                    CStr(_) | ChanMat(_) -> ref.channel
                  | ChanRef(c) -> c.channel
                  | _ as t -> (raise (Failure("Cannot assign "^
                                              (string_of_vdecl (type_of_expr
                                              env1 t))^" to "^
                                              ref.image^":"^ref.channel)))
              in
              let get_chandef = function
```

```
                   CStr(s,idl) -> true, (s,idl), false, ((BInt(1),BInt(1)),[[BInt(1)]])
                 | ChanMat(m) -> true, ("",[]), true, m
                 | _ -> false, ("",[]), false, ((BInt(1),BInt(1)),[[BInt(1)]])
               in
               let channame = get_channame ve in
               let isvalid, cfuncstr, ismat, cmat = get_chandef ve in
               let ctype = get_calctype env1 channame in
               (* Add channel to image! *)
               let env2 = image_add env1 (find_image env ref.image) ref.channel
                                 ctype isvalid ismat cfuncstr cmat
               in env2
              else
                  (raise (Failure(s)));
          )
        in envNew, ChanAssign(ref,ve)
  | LibCall(f,args) -> check_libf env f args

and check_libf env libf args =
  match libf with
      ImgRead  -> check_imgRead env args
    | ImgWrite -> check_imgWrite env args

and check_imgRead env args =
  match args with
      hd :: [] -> let env1, file = check_expr env hd in
                    (match file with
                        LitStr(s) -> env1, LibCall(ImgRead, [file])
                      | Integer(bi) -> env1, LibCall(ImgRead, [file])
                      | _ -> raise(Failure("ImgRead parameter must be a string or an integer"))
                    )
    | _ -> raise(Failure("ImgRead must have exactly one argument"))

and check_imgWrite env args =
  match args with
      raw_fname_expr :: raw_fmt_expr :: raw_img_expr :: [] -> (
        (* 1 Check fname_expr is string literal or an integer *)
        (* 2 Check fmt_expr is string literal *)
        (* 3 Check img_expr is a valid image expression *)
        let fname_expr = match raw_fname_expr with
            LitStr(s) -> LitStr(s)
          | Integer(bi) -> Integer(bi)
          | _ -> raise(Failure("3rd parameter of ImgWrite must be a string or an integer"))
        in
        let fmt_expr = match raw_fmt_expr with
            LitStr(s) -> LitStr(s)
          | _ -> raise(Failure("2nd parameter of ImgWrite must be a string representing the format"))
        in
        let env1, img_expr = check_expr env raw_img_expr in
        let _ = match type_of_expr env1 img_expr with
                    ImageT(s) -> ()
                  | _ -> raise(Failure("The validator is not pleased: "^
                                   "ImgWrite's 1st arg is NOT an image identifier!"))
        in
        env1, LibCall(ImgWrite, [fname_expr; fmt_expr; img_expr])
      )
    | _ -> raise(Failure("Wrong number of arguments supplied to ImgWrite function"))


(*
 * VERIFY:
 *   our statement checker (invokes the expression checker)
 *)
let check_stmt env = function
    Expr(e) -> let env1, vexpr = check_expr env e in
          env1, Expr(vexpr)
  | VDecl(v) -> let env1 = var_add env v in
          env1, VDecl(v)
  | VAssign(v,op,e) ->
        (* NOTE: order is important here!
         *       the variable is not declared "in scope" until the end
         *       of the statement (i.e. you can't reference the variable
         *       from within its assignment inititalizer)
         *)
        let env1, vexpr = check_expr env e in
        let env2 = var_add env v in
        let lhs = v in
        let rhs = type_of_expr env2 vexpr in
        let env3 = check_assignment env2 rhs vexpr op lhs in
        env3, VAssign(v, op, vexpr)

(*
 * VERIFIER!
 *)
let verify program =
  let venv, vslist = (List.fold_left
    (fun (env0,slist) s -> let env, vstmt =
                             check_stmt env0 s in env, vstmt :: slist)
    ( {calc = [];
       conv = [];
       images = [];
       kernels = [];
       allvars = [];}, [] ) program)
  in
  venv, (List.rev vslist)
```

A.1.1 Subset of *extlib* Used by CLAM

CLAM compiled the following source files from the `extlib` [3] project. Their full source is omitted for brevity – see the project website.

- `enum.ml`
- `enum.mli`
- `extString.ml`
- `extString.mli`
- `std.ml`
- `std.mli`

A.2 CLAM Implementation Library (`clam.a`)

```
#ifndef CLAM_H
#define CLAM_H
/*
 * CLAM C Interface Header
 *
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include <float.h>
#include <limits.h>
#include <math.h>

#define __STDC_LIMIT_MACROS
#include <stdint.h>


/* --- --- --- --- --- --- --- */
/* stolen from: linux/list.h   */
/* --- --- --- --- --- --- --- */

#ifdef __cplusplus
extern "C" {
#endif

#ifdef _DEBUG
#define DBG(X) X
#else
#define DBG(X)
#endif

struct list_head {
        struct list_head *next, *prev;
};

static inline void INIT_LIST_HEAD(struct list_head *nm)
{
        nm->next = nm;
        nm->prev = nm;
}

static inline void __list_add(struct list_head *newI,
                              struct list_head *prev,
                              struct list_head *next) {
        next->prev = newI;
        newI->next = next;
        newI->prev = prev;
        prev->next = newI;
}
```

```

static inline void list_add(struct list_head *newI, struct list_head *head)
{
        __list_add(newI, head, head->next);
}

static inline void list_add_tail(struct list_head *newI, struct list_head *head)
{
        __list_add(newI, head->prev, head);
}

#define __list_del(__prev, __next) \
        (__next)->prev = __prev; \
        (__prev)->next = __next

static inline void list_del(struct list_head *entry)
{
        __list_del(entry->prev, entry->next);
}

static inline void list_del_init(struct list_head *entry)
{
        __list_del(entry->prev, entry->next);
        INIT_LIST_HEAD(entry);
}

static inline void list_move(struct list_head *list, struct list_head *head)
{
        __list_del(list->prev, list->next);
        list_add(list, head);
}

static inline void list_move_tail(struct list_head *list,
                                  struct list_head *head)
{
        __list_del(list->prev, list->next);
        list_add_tail(list, head);
}

static inline int list_is_last(const struct list_head *list,
                                const struct list_head *head)
{
        return list->next == head;
}

static inline int list_empty(const struct list_head *head)
{
        return head->next == head;
}

#ifndef offsetof
#define offsetof(st, m) \
     ((size_t) ( (char *)&((st *)(0))->m - (char *)0 ))
#endif

#define container_of(ptr, type, member) ({                      \
        const typeof( ((type *)0)->member ) *__mptr = (ptr);    \
        (type *)( (char *)__mptr - offsetof(type,member) );})

#define list_entry(ptr, type, member) \
        container_of(ptr, type, member)

#define list_first_entry(ptr, type, member) \
        list_entry((ptr)->next, type, member)

#define list_for_each(pos, head) \
        for (pos = (head)->next; pos != (head); pos = pos->next)

#define list_for_each_prev(pos, head) \
        for (pos = (head)->prev; pos != (head); pos = pos->prev)

#define list_for_each_safe(pos, n, head) \
        for (pos = (head)->next, n = pos->next; pos != (head); \
                pos = n, n = pos->next)

#define list_for_each_prev_safe(pos, n, head) \
        for (pos = (head)->prev, n = pos->prev; \
             pos != (head); \
             pos = n, n = pos->prev)

#define list_for_each_entry(pos, head, member)                          \
        for (pos = list_entry((head)->next, typeof(*pos), member);      \
             &pos->member != (head);    \
             pos = list_entry(pos->member.next, typeof(*pos), member))

#define list_for_each_entry_reverse(pos, head, member)                  \
        for (pos = list_entry((head)->prev, typeof(*pos), member);      \
             &pos->member != (head);    \
             pos = list_entry(pos->member.prev, typeof(*pos), member))


/* --- --- --- --- --- --- --- */
/* CLAM type declarations      */
/* --- --- --- --- --- --- --- */

typedef enum clam_img_fmt_e {
        PNG = 0,
        BMP,
        TGA,
        JPG,
        CLAM_NUMFMTS,
} clam_img_fmt;


typedef enum clam_atom_e {
        UINT8 = 0,
        UINT16,
        UINT32,
        INT8,
        INT16,
        INT32,
        ANGLE,
        CLAM_NUMTYPES,
} clam_atom;

#define clam_type_union \
        uint8_t  u8;  \
        uint16_t u16; \
        uint32_t u32; \
        int8_t   s8;  \
        int16_t  s16; \
        int32_t  s32; \
        float    f32;

static inline int clam_atom_sz(clam_atom a) {
        switch (a) {
        case UINT8:
                return sizeof(uint8_t);
        case UINT16:
                return sizeof(uint16_t);
        case UINT32:
                return sizeof(uint32_t);
        case INT8:
                return sizeof(int8_t);
        case INT16:
                return sizeof(int16_t);
        case INT32:
                return sizeof(int32_t);
        case ANGLE:
                return sizeof(float);
        default:
                return 0;
        }
}

/* ImageT */
typedef struct clam_img {
        unsigned char *p;
        int width;
        int height;
        struct list_head chan;   /* master channel list */
        int num_chan;            /* total number of channels */
        unsigned char **curr_p;  /* channel data pointers: dynamically setup */
        unsigned int   *curr_s;
        const char *name;
} clam_img;

/* clam matrix (for kernel computation) */
typedef struct clam_matrix {
        int32_t rows, cols;
        union { clam_type_union } num;
        union { clam_type_union } denom;
        void *d;
} clam_matrix;

#define clam_calc_setmatrix(_calc, _type, _rows, _cols, _num, _denom, _data...) \
        (_calc)->ismat = 1; \
        (_calc)->m.rows = _rows; \
        (_calc)->m.cols = _cols; \
        *((_type *)(&(_calc)->m.num)) = _num; \
        *((_type *)(&(_calc)->m.denom)) = _denom; \
        static int _calc ## _matdata [_rows][_cols] = _data; \
        (_calc)->m.d = & _calc ## _matdata

/* CalcT */
typedef struct clam_calc {
        const char  *name;
        clam_atom    type;
        int          ismat;
        clam_matrix  m;
} clam_calc;

/* elements of a KernelT */
typedef struct clam_kcalc {
        struct list_head list;
        int used;
        clam_calc *calc;
} clam_kcalc;

/* KernelT */
typedef struct clam_kernel {
        struct list_head allcalc;
} clam_kernel;

/* ImageT channels */
typedef struct clam_imgchan {
        struct list_head  list;
        clam_img         *img;
        const char       *name;
        unsigned char    *p;
        clam_atom         type;
        uint32_t          stride;
} clam_imgchan;


/* --- --- --- --- --- --- --- */
/* internal (compiler) API     */
/* --- --- --- --- --- --- --- */

#define bail(msg, ...) \
{ \
        fprintf(stderr, "CLAM Runtime ERROR: " msg "\n", ## __VA_ARGS__ ); \
        exit(EXIT_FAILURE); \
}

#define clam_alloc_check(var) \
        if (!var) bail("out of memory for " #var)

static inline clam_img *clam_img_alloc(void)
{
        clam_img *img;
        img = (clam_img *)malloc(sizeof(*img));
        if (!img)
                return NULL;
        /* simple init */
        memset(img, 0, sizeof(*img));
        img->width = img->height = -1;
        img->num_chan = 0;
        INIT_LIST_HEAD(&img->chan);

        return img;
}

static inline void clam_img_free(clam_img *img)
{
        struct list_head *pos, *tmp;
        clam_imgchan *ch;

        if (!img) return;
        list_for_each_safe(pos, tmp, &img->chan) {
                ch = list_entry(pos, typeof(*ch), list);
                list_del(pos);
                free(ch->p);
                free(ch);
        }
        free(img->curr_p);
        free(img->curr_s);
        free(img->p);
        free(img);
}

static inline int clam_img_valid(clam_img *img)
{
        return (img->width > 0) && (img->height > 0);
}

static inline void clam_img_setup_calc(clam_img *img)
{
        clam_imgchan *ch;
        int i;

        free(img->curr_p); /* kill previous setup */
        free(img->curr_s);
        img->curr_p = (unsigned char **)malloc(img->num_chan * sizeof(char *));
        img->curr_s = (unsigned int *)malloc(img->num_chan * sizeof(int));
        if (!img->curr_p || !img->curr_s) {
                fprintf(stderr, "Internal memory alloc error\n");
                return;
        }

        i = 0;
        list_for_each_entry(ch, &img->chan, list) {
                img->curr_p[i] = ch->p;
                img->curr_s[i] = ch->stride;
                i++;
        }
}

#define clam_img_next_pix(img) \
        { int __chidx__; \
        for (__chidx__ = 0; __chidx__ < img->num_chan; __chidx__++) { \
                img->curr_p[__chidx__] += img->curr_s[__chidx__]; \
        } \
        }

#define clam_img_pix(type, chidx) \
        (*((type *)((pp)[chidx])))

static inline clam_calc *clam_calc_alloc(const char *name, clam_atom type)
{
        clam_calc *c;
        c = (clam_calc *)malloc(sizeof(*c));
        clam_alloc_check(c);
        memset(c, 0, sizeof(*c));
        c->name = name;
        c->type = type;
        return c;
}

static inline clam_kernel *clam_kernel_alloc(void)
{
        clam_kernel *k;
        k = (clam_kernel *)malloc(sizeof(*k));
        clam_alloc_check(k);
        memset(k, 0, sizeof(*k));
        INIT_LIST_HEAD(&k->allcalc);
        return k;
}

static inline void clam_kernel_free(clam_kernel *kern)
{
        struct list_head *pos, *tmp;
        clam_kcalc *kc;
        if (!kern) return;

        list_for_each_safe(pos, tmp, &kern->allcalc) {
                kc = list_entry(pos, typeof(*kc), list);
                list_del(pos);
                free(kc);
        }

        free(kern);
}

```

```
#define TEMP_clam_kernel_addcalc(K,C) \
        clam_kernel_addcalc(K, C, 1)

static clam_kernel *clam_kernel_addcalc(clam_kernel *kern, clam_calc *calc, int used)
{
        clam_kcalc *kc;
        kc = (clam_kcalc *)malloc(sizeof(*kc));
        if (!kc) {
                fprintf(stderr, "out of memory\n");
                return NULL;
        }
        memset(kc, 0, sizeof(*kc));
DBG(    printf("Adding %s to kernel\n", calc->name);)
        INIT_LIST_HEAD(&kc->list);
        kc->calc = calc;
        kc->used = used;
        list_add(&kc->list, &kern->allcalc);

        return kern;
}

/* --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- */
/*                                                                 */
/* CLAM heavy lifting functions                                    */
/* (implemented in generated C file)                               */
/*                                                                 */
/* --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- */

extern clam_img *clam_img_copy(clam_img *src);

#define clam_img_assign(DST, SRC) \
        ({ if (DST) clam_img_free(DST); DST = (SRC); })

extern clam_kernel *clam_kernel_copy(clam_kernel *src);

#define clam_kernel_assign(DST, SRC) \
        ({ if (DST) clam_kernel_free(DST); DST = (SRC); })

extern clam_img *__clam_imgchan_add(clam_img *img, clam_atom type,
                                    const char *name, int should_alloc);

#define clam_imgchan_addcalc(IMG, CHAN) \
        __clam_imgchan_add(IMG, (CHAN)->type, #CHAN, 0)

#define clam_imgchan_add_empty(IMG, NAME, TYPE) \
        __clam_imgchan_add(IMG, TYPE, #NAME, 1)

extern void clam_imgchan_del(clam_img *img, const char *name);

extern int clam_imgchan_exists(clam_img *img, const char *name);

#define clam_imgchan_ref(img, name) \
        __clam_imgchan_ref(img, name, NULL)

extern clam_imgchan *__clam_imgchan_ref(clam_img *img, const char *name, int *idx);

#define clam_imgchan_copy(DST, DNAME, SCHAN) \
        ({if (!DST) DST = clam_img_alloc(); \
        __clam_imgchan_copy(DST, DNAME, SCHAN); })

extern clam_imgchan *__clam_imgchan_copy(clam_img *dst, const char *dname,
                                         clam_imgchan *schan);

extern void clam_img_resize(clam_img *img, int width, int height);

/* Functional Library Interface */
extern clam_img *imgread(const char *filename);
extern int imgwrite(clam_img *img, clam_img_fmt fmt, const char *filename);

#define clam_imgchan_eval(__img, __type, __ch) \
{ \
        int pix, sz; \
        unsigned char *chan_ptr; \
        unsigned char **pp; \
        __type *val; \
        if (!((__ch)->p)) { \
                sz = (__img)->width * (__img)->height; \
                val = (__type *)malloc(sz * (__ch)->stride); \
                clam_alloc_check(val); \
                (__ch)->p = (unsigned char *)val; \
                clam_img_setup_calc(__img); \
                for (pix = 0; pix < sz; ++pix) { \
                        pp = (__img)->curr_p; \
                        *val++ = (__type)( cfunc ); \
                        clam_img_next_pix(__img); \
                } \
        } \
}

#define clam_convolve_cfunc(CALC,TYPE,CFUNC...) \
{ \
        clam_imgchan *__outchanref; \
        __clam_imgchan_add(__IMG, (CALC)->type, (CALC)->name, 0); \
        __outchanref = clam_imgchan_ref(__IMG, (CALC)->name); \
        clam_imgchan_eval(__IMG, TYPE, __outchanref); \
}

/* Declare a function for every convolution instance in the program */
#define clam_convfunc_start(IDX,IMGNAME,CHANNAME) \
clam_img *__convolution ## IDX(clam_kernel *kern) { \
        clam_kcalc *__kc; \
        clam_img *__IMG; \
        clam_imgchan *__CONVCHAN = clam_imgchan_ref(IMGNAME, #CHANNAME); \
        __IMG = clam_img_alloc(); \
        list_for_each_entry_reverse(__kc, &kern->allcalc, list) { \
                clam_calc *__c = __kc->calc; \
                if (__c->ismat) { \
                        clam_convolve_matrix(__IMG, __CONVCHAN, __c); \
                } else { \

/* if this channel is associated with a CString, then we have to run
 * that calculation: the CLAM backend will spit out a do_CHANNAME
 * label to which we will jump (and assume the label is nice and jumps
 * back)... it's kind of like a dumb function call... we do it this way
 * because the backend also spits out a bunch of pre-processor tokens.
 * ugh.
 */
#define clam_convfunc_chk(CHAN) \
                        if (strcmp(__c->name, #CHAN) == 0) \
                                goto do_ ## CHAN;

#define clam_convfunc_lastchk() \
                        continue;

#define clam_convfunc_end(IDX) \
                } \
        } \
        clam_img_cleanup(__IMG, kern); \
        return __IMG; \
}


#ifdef __cplusplus
} /* extern "C" */

extern void clam_convolve_matrix(clam_img *outimg,
                                 clam_imgchan *ch,
                                 clam_calc *calc);

/* Convolution (much easier with templates) */
template<typename CalcT, typename ChanT>
void __clam_convolve_matrix(clam_img *outimg, clam_imgchan *ch, clam_calc *calc, int min, int max)
{
        int xx, xstart, xend;
        int yy, ystart, yend;
        int kx, kxstart, kxend;
        int ky, kystart, kyend;
        int k_start, k_stride;
        int width, height;
        clam_imgchan *outchan;
        CalcT *dpix;
        ChanT *spix;

        int *kern, num, denom;

        if (!calc->ismat)
                return;

        if (!clam_img_valid(outimg))
                clam_img_resize(outimg, ch->img->width, ch->img->height);

        /* add the channel to the image
```

```
	 * (if it's unused, we'll remove it later)
	 */
	__clam_imgchan_add(outimg, calc->type, calc->name, 1);
	outchan = clam_imgchan_ref(outimg, calc->name);

	width = outimg->width;
	height = outimg->height;

	kern = (int *)(calc->m.d);
	num = (int)*((CalcT *)&(calc->m.num));
	denom = (int)*((CalcT *)&(calc->m.denom));

DBG(	printf("\tConvolve[%d/%d]: %s:%s = %s ** %s\n", num,denom, outimg->name, calc->name, ch->name,
     calc->name);)
	/* XXX: remove this when we add boundary support! */
	memset(outchan->p, 0, width * height * sizeof(CalcT));

	/* XXX: do first N rows */

	ystart = calc->m.rows/2 + 1;
	yend = height - calc->m.rows;
	xstart = calc->m.cols/2 + 1;
	xend = width - calc->m.cols;

	kystart = -(calc->m.rows/2);
	kyend = kystart + calc->m.rows;
	kxstart = -(calc->m.cols/2);
	kxend = kxstart + calc->m.cols;

	k_start = kystart*width + kxstart;
	k_stride = width - calc->m.cols;

	for (yy = ystart; yy < yend; yy++) {
		/* XXX: do first N cols */

		dpix = &( ((CalcT *)outchan->p)[(yy * width) + xstart] );
		spix = &( ((ChanT *)ch->p)[(yy * width) + xstart]);
		for (xx = xstart; xx < xend; xx++, dpix++, spix++) {
			/* kernel operation */
			int kidx = 0;
			int sidx = k_start;
			int val = 0;
			for (ky = kystart; ky < kyend; ky++, sidx += k_stride) {
				for (kx = kxstart; kx < kxend; kx++, kidx++, sidx++) {
					val += (int)(spix[sidx]) * kern[kidx];
				}
			}
			val *= num;
			val /= denom;
			/* clamp */
			val = (val < min ? min : (val > max ? max : val));
			*dpix = (CalcT)(val);
		}

		/* XXX: do last N cols */
	}

	/* XXX: do last N rows */
}

/* XXX: create a partially specialized function for float... ?! */
#endif

#endif /* CLAM_H */
```

```
/*
 * Template C file for CLAM backend
 * Jeremy C. Andrus <jeremya@cs.columbia.edu>
 * 2011-12-12
 */

/* --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- */
/*                                                                  */
/* CLAM heavy lifting functions                                     */
/*                                                                  */
/* --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- */

clam_img *clam_img_copy(clam_img *src)
{
        bail("image copy not quite supported... check back later");
}

clam_kernel *clam_kernel_copy(clam_kernel *src)
{
        bail("kernel copy not quite supported... check back later");
}

clam_img *__clam_imgchan_add(clam_img *img, clam_atom type,
                             const char *name, int should_alloc)
{
        clam_imgchan *chan;

DBG(    printf("Adding %s to %s\n", name, img->name);)
        chan = (clam_imgchan *)malloc(sizeof(*chan));
        if (!chan)
                bail("no memory for channel");

        INIT_LIST_HEAD(&chan->list);
        chan->name = name;
        chan->img = img;
        chan->type = type;
        chan->stride = clam_atom_sz(type);
        if (should_alloc) {
                int sz = img->width * img->height * chan->stride;
                chan->p = (unsigned char *)malloc(sz);
                if (!chan->p)
                        bail("no memory for channel");
        } else
                chan->p = NULL;

        list_add_tail(&chan->list, &img->chan);
        img->num_chan++;
        return img;
}

void clam_imgchan_del(clam_img *img, const char *name)
{
DBG(    printf("Removing %s from %s\n", name, img->name);)
        clam_imgchan *ch = clam_imgchan_ref(img, name);
        list_del(&ch->list);
        free(ch->p);
        free(ch);
}

int clam_imgchan_exists(clam_img *img, const char *name)
{
        clam_imgchan *ch;
        list_for_each_entry(ch, &img->chan, list) {
                if (strcmp(name, ch->name) == 0)
                        return 1;
        }
        return 0;
}


clam_imgchan *__clam_imgchan_ref(clam_img *img, const char *name, int *idx)
{
        int ii = 0;
        clam_imgchan *ch;
        list_for_each_entry(ch, &img->chan, list) {
                if (strcmp(name, ch->name) == 0) {
                        goto out;
                }
                ii++;
        }
        bail("Invalid channel: %s", name);
out:
        if (idx) *idx = ii;
        return ch;
}

clam_imgchan *__clam_imgchan_copy(clam_img *dst, const char *dname,
                                  clam_imgchan *schan)
{
        int sz;
        clam_imgchan *dchan;
        clam_img *src = schan->img;

DBG(    printf("Copy-> %s:%s = %s:%s\n",dst->name,dname,schan->img->name,schan->name);)
        if (!clam_img_valid(dst))
                clam_img_resize(dst, src->width, src->height);

        if (dst->width != src->width || dst->height != src->height) {
                bail("incompatible images in chan copy (%s->%s)", schan->name, dname);
        }

        if (clam_imgchan_exists(dst, dname))
                clam_imgchan_del(dst, dname);

        __clam_imgchan_add(dst, schan->type, dname, 1);
        dchan = clam_imgchan_ref(dst, dname);

        sz = src->width * src->height * schan->stride;
        memcpy(dchan->p, schan->p, sz);
}

void clam_img_resize(clam_img *img, int width, int height)
{
        if (!list_empty(&img->chan))
                bail("Can't resize an image with existing channels!");

        img->width = width;
        img->height = height;
        img->num_chan = 0;
        free(img->curr_p); img->curr_p = NULL;
        free(img->curr_s); img->curr_s = NULL;
}

/* The ugly ugly dispatch function... this is the price you pay for making
 * all your types just "magically" work together... */
void clam_convolve_matrix(clam_img *outimg,
                          clam_imgchan *ch,
                          clam_calc *calc)
{
        /* switch on destination type (CalcT) */
        switch (calc->type) {
        case UINT8:
                /* switch on source type (ChanT) */
                switch (ch->type) {
                case UINT8:
                        __clam_convolve_matrix<uint8_t, uint8_t>(outimg, ch, calc, 0, UINT8_MAX);
                        break;
                case UINT16:
                        __clam_convolve_matrix<uint8_t, uint16_t>(outimg, ch, calc, 0, UINT8_MAX);
                        break;
                case UINT32:
                        __clam_convolve_matrix<uint8_t, uint32_t>(outimg, ch, calc, 0, UINT8_MAX);
                        break;
                case INT8:
                        __clam_convolve_matrix<uint8_t, int8_t>(outimg, ch, calc, 0, UINT8_MAX);
                        break;
                case INT16:
                        __clam_convolve_matrix<uint8_t, int16_t>(outimg, ch, calc, 0, UINT8_MAX);
                        break;
                case INT32:
                        __clam_convolve_matrix<uint8_t, int32_t>(outimg, ch, calc, 0, UINT8_MAX);
                        break;
                case ANGLE:
                        __clam_convolve_matrix<uint8_t, float>(outimg, ch, calc, 0, UINT8_MAX);
                        break;
                default:
                        bail("invalid channel type?!");
                }
                break;
        case UINT16:
                switch (ch->type) {
```

```
                case UINT8:
                        __clam_convolve_matrix<uint16_t, uint8_t>(outimg, ch, calc, 0, UINT16_MAX);
                        break;
                case UINT16:
                        __clam_convolve_matrix<uint16_t, uint16_t>(outimg, ch, calc, 0, UINT16_MAX);
                        break;
                case UINT32:
                        __clam_convolve_matrix<uint16_t, uint32_t>(outimg, ch, calc, 0, UINT16_MAX);
                        break;
                case INT8:
                        __clam_convolve_matrix<uint16_t, int8_t>(outimg, ch, calc, 0, UINT16_MAX);
                        break;
                case INT16:
                        __clam_convolve_matrix<uint16_t, int16_t>(outimg, ch, calc, 0, UINT16_MAX);
                        break;
                case INT32:
                        break;
                        __clam_convolve_matrix<uint16_t, int32_t>(outimg, ch, calc, 0, UINT16_MAX);
                case ANGLE:
                        break;
                        __clam_convolve_matrix<uint16_t, float>(outimg, ch, calc, 0, UINT16_MAX);
                default:
                        bail("invalid channel type?!");
                }
                break;
        case UINT32:
                switch (ch->type) {
                case UINT8:
                        __clam_convolve_matrix<uint32_t, uint8_t>(outimg, ch, calc, 0, UINT32_MAX);
                        break;
                case UINT16:
                        __clam_convolve_matrix<uint32_t, uint16_t>(outimg, ch, calc, 0, UINT32_MAX);
                        break;
                case UINT32:
                        __clam_convolve_matrix<uint32_t, uint32_t>(outimg, ch, calc, 0, UINT32_MAX);
                        break;
                case INT8:
                        __clam_convolve_matrix<uint32_t, int8_t>(outimg, ch, calc, 0, UINT32_MAX);
                        break;
                case INT16:
                        __clam_convolve_matrix<uint32_t, int16_t>(outimg, ch, calc, 0, UINT32_MAX);
                        break;
                case INT32:
                        __clam_convolve_matrix<uint32_t, int32_t>(outimg, ch, calc, 0, UINT32_MAX);
                        break;
                case ANGLE:
                        __clam_convolve_matrix<uint32_t, float>(outimg, ch, calc, 0, UINT32_MAX);
                        break;
                default:
                        bail("invalid channel type?!");
                }
                break;
        case INT8:
                switch (ch->type) {
                case UINT8:
                        __clam_convolve_matrix<int8_t, uint8_t>(outimg, ch, calc, INT8_MIN, INT8_MAX);
                        break;
                case UINT16:
                        __clam_convolve_matrix<int8_t, uint16_t>(outimg, ch, calc, INT8_MIN, INT8_MAX);
                        break;
                case UINT32:
                        __clam_convolve_matrix<int8_t, uint32_t>(outimg, ch, calc, INT8_MIN, INT8_MAX);
                case INT8:
                        break;
                        __clam_convolve_matrix<int8_t, int8_t>(outimg, ch, calc, INT8_MIN, INT8_MAX);
                case INT16:
                        break;
                        __clam_convolve_matrix<int8_t, int16_t>(outimg, ch, calc, INT8_MIN, INT8_MAX);
                case INT32:
                        break;
                        __clam_convolve_matrix<int8_t, int32_t>(outimg, ch, calc, INT8_MIN, INT8_MAX);
                case ANGLE:
                        break;
                        __clam_convolve_matrix<int8_t, float>(outimg, ch, calc, INT8_MIN, INT8_MAX);
                default:
                        bail("invalid channel type?!");
                }
                break;
        case INT16:
                switch (ch->type) {
                case UINT8:
```

```
                                __clam_convolve_matrix<int16_t, uint8_t>(outimg, ch, calc, INT16_MIN, INT16_MAX);
                                break;
                        case UINT16:
                                __clam_convolve_matrix<int16_t, uint16_t>(outimg, ch, calc, INT16_MIN, INT16_MAX)
                                    ;
                                break;
                        case UINT32:
                                __clam_convolve_matrix<int16_t, uint32_t>(outimg, ch, calc, INT16_MIN, INT16_MAX)
                                    ;
                                break;
                        case INT8:
                                __clam_convolve_matrix<int16_t, int8_t>(outimg, ch, calc, INT16_MIN, INT16_MAX);
                                break;
                        case INT16:
                                __clam_convolve_matrix<int16_t, int16_t>(outimg, ch, calc, INT16_MIN, INT16_MAX);
                                break;
                        case INT32:
                                __clam_convolve_matrix<int16_t, int32_t>(outimg, ch, calc, INT16_MIN, INT16_MAX);
                                break;
                        case ANGLE:
                                __clam_convolve_matrix<int16_t, float>(outimg, ch, calc, INT16_MIN, INT16_MAX);
                                break;
                        default:
                                bail("invalid channel type?!");
                        }
                        break;
                case INT32:
                        switch (ch->type) {
                        case UINT8:
                                __clam_convolve_matrix<int32_t, uint8_t>(outimg, ch, calc, INT32_MIN, INT32_MAX);
                                break;
                        case UINT16:
                                __clam_convolve_matrix<int32_t, uint16_t>(outimg, ch, calc, INT32_MIN, INT32_MAX)
                                    ;
                                break;
                        case UINT32:
                                __clam_convolve_matrix<int32_t, uint32_t>(outimg, ch, calc, INT32_MIN, INT32_MAX)
                                    ;
                                break;
                        case INT8:
                                __clam_convolve_matrix<int32_t, int8_t>(outimg, ch, calc, INT32_MIN, INT32_MAX);
                                break;
                        case INT16:
                                __clam_convolve_matrix<int32_t, int16_t>(outimg, ch, calc, INT32_MIN, INT32_MAX);
                                break;
                        case INT32:
                                __clam_convolve_matrix<int32_t, int32_t>(outimg, ch, calc, INT32_MIN, INT32_MAX);
                                break;
                        case ANGLE:
                                __clam_convolve_matrix<int32_t, float>(outimg, ch, calc, INT32_MIN, INT32_MAX);
                                break;
                        default:
                                bail("invalid channel type?!");
                        }
                        break;
                case ANGLE:
                        switch (ch->type) {
                        case UINT8:
                                __clam_convolve_matrix<float, uint8_t>(outimg, ch, calc, 0, INT32_MAX);
                                break;
                        case UINT16:
                                __clam_convolve_matrix<float, uint16_t>(outimg, ch, calc, 0, INT32_MAX);
                                break;
                        case UINT32:
                                __clam_convolve_matrix<float, uint32_t>(outimg, ch, calc, 0, INT32_MAX);
                                break;
                        case INT8:
                                __clam_convolve_matrix<float, int8_t>(outimg, ch, calc, 0, INT32_MAX);
                                break;
                        case INT16:
                                __clam_convolve_matrix<float, int16_t>(outimg, ch, calc, 0, INT32_MAX);
                                break;
                        case INT32:
                                __clam_convolve_matrix<float, int32_t>(outimg, ch, calc, 0, INT32_MAX);
                                break;
                        case ANGLE:
                                __clam_convolve_matrix<float, float>(outimg, ch, calc, 0, INT32_MAX);
                                break;
                        default:
                                bail("invalid channel type?!");
                        }
                break;
        default:
                bail("invalid calculation type?!");
        }
}


void clam_img_cleanup(clam_img *img, clam_kernel *kern)
{
        clam_kcalc *kc;
        list_for_each_entry(kc, &kern->allcalc, list) {
                if (!kc->used)
                        clam_imgchan_del(img, kc->calc->name);
        }
}
```

A.3 Unit Test Framework

Our unit testing framework was built on two custom shell scripts that provided a framework to compile a test program, determine success/failure of compilation, compare image outputs and report overall success or failure of the test. The framework simply runs all shell scripts in the *test* directory and reports the success/failure of each one (with a summary of failures at the end). The `_buildup.sh`, `_breakdown.sh`, and `all.test` scripts are reported here, followed by all of the tests and a sample run of the `all.test` script.

_buildup.sh

```
#!/bin/bash
# _buildup.sh

TEST_NAME=$(basename ${0%\.*})
TEST_SRC=${TEST_NAME}.clam
TEST_BIN=${TEST_NAME}.bin

function msgok() { echo -e "\033[00;34mOK $1\033[00m"; }

function error() { echo -e "\033[00;33mERROR $1\033[00m"; }

# Check that necessary unit test variables exist
if [ -z "$TEST_DESC" ]; then
        error "Unit test not properly formatted. Description required!"
        exit 1
fi

# Check if we're asking for help. If so, provide usage
if [ $# == 1 ] && ([ "$1" == "-help" ] || [ "$1" == "--help" ] || [ "$1" == "-h" ]); then
        echo "CLAM Unit Test: $TEST_NAME"
        echo "Description: $TEST_DESC"
        echo "Usage: $0 [optional/path/to/clam/binary]"
        exit 0
fi

# Find CLAM, verify it exists
CLAM_BINARY=${1:-../clam}
if [ ! -x $CLAM_BINARY ]; then
        error "Could not find CLAM binary at: $CLAM_BINARY"
        exit 1
fi

function compare() {
        cmp -s "$1" "$2"
        return $?
}

function compile_it() {
        if [ ! -f "$TEST_SRC" ]; then
                error "Could not find $TEST_NAME source file: '$TEST_SRC'"
                exit 1
        fi
        echo -n "Compiling '$TEST_SRC'..."
        COMPILE_OUTPUT=$(${CLAM_BINARY} -i $TEST_SRC -o ./$TEST_BIN 2>&1)
        ERRORS=$?
        if [ ! $ERRORS -eq 0 ]; then error "${COMPILE_OUTPUT#Error: }"; else msgok "."; fi
}

function run_it() {
        if [ ! -x "$TEST_BIN" ]; then
                RUN_OUTPUT="Missing binary!"
                ERRORS=1
        fi
        RUN_OUTPUT=$(./$TEST_BIN $@ 2>&1)
        ERRORS=$?
}

echo "======================================="
echo "** $TEST_NAME"
echo "** $TEST_DESC"

# Make sure a test can't pass by accident
ERRORS=100000
```

_breakdown.sh

```
#!/bin/bash
# _breakdown.sh

rm -f ${TEST_NAME}.bin 2>&1

echo "----------------------------------------"
if [ $ERRORS -eq 0 ]; then
        echo -e "$TEST_NAME: [ \033[00;32mPASS\033[00m ]"
        echo
        exit 0
else
        echo -e "$TEST_NAME: [ \033[00;31mFAIL\033[00m ]"
        echo
        exit 1
fi
```

all.test

```
#!/bin/bash
DIR="$( cd "$( dirname "$0" )" && pwd )"
cd $DIR
echo "PWD: $PWD"

CNT=0
PASS=0
FAILING=""
for t in *.test; do
        if [ "$(basename $0)" == "$(basename $t)" ]; then continue; fi
        CNT=$(($CNT + 1))
        ./$t
        if [ $? -eq 0 ]; then
                PASS=$(($PASS + 1))
        else
                NM=$(basename $t)
                FAILING="$FAILING ${NM%.test}"
        fi
done

echo
echo "PASSED: $PASS / $CNT"
if [ ! -z "$FAILING" ]; then
  echo "FAILED: $FAILING"
fi
echo
```

The tests (shell script followed by CLAM source)

Unit Test: 1calc-ker

```
#!/bin/bash
TEST_DESC="test a kernel with 1 calc" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "cannot declear a kernel with one Calc"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };

Kernel sobel = | sobelGx;
```

Unit Test: DefEq

```
#!/bin/bash
TEST_DESC="Using Eq instead of DefEq to define a Calc"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Calc is defined by a Eq instead of DefEq"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Calc test1<Uint8> = [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
```

Unit Test: addker

```
#!/bin/bash
TEST_DESC="Test using OrEq to add kernal to a Kernel" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "cannot add a kernal to another by OrEq"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image imtest;

Calc c1<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc c2<Uint8> :=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel k1 = | c1 | c2;
Kernel k2 = | c2 | c1;

k1 |= k2;
```

Unit Test: at-channel

```
#!/bin/bash
TEST_DESC="Test @operator"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Accessed channel marked with @!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Image srcimg = imgread("ucla.png");

Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
Calc sobelG<Uint8>:= #[sqrt(sobelGx*sobelGx + sobelGy*sobelGy)]#;
Calc sobelTheta<Angle>:= #[arctan(sobelGy/sobelGx)]#;

srcimg |= Lum;

Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | @sobelGx | @sobelGy | sobelG;
sobel |= sobelTheta;

Image edges = srcimg:Lum ** sobel;
Image output;
output:Red   = edges:sobelGx; /* This should be bad */
```

Unit Test: cimage

```
#!/bin/bash
TEST_DESC="Test defining an image with warp C"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Define an image with warp C"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
/* should not allow this: */
Image img = #[23]#;
```

Unit Test: ckernel

```
#!/bin/bash
TEST_DESC="Test defining a kernal with one or more warp C"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Define a kernal with warp C"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Kernel t1 = #[Red*2]# | #[Blue]#;
```

Unit Test: comment1

```
#!/bin/bash
TEST_DESC="Write an empty program with a comment and make sure it works."
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        run_it
        ERRORS=$?
        if [ $ERRORS -ne 0 ]; then
                error "Binary did not exit normally: $ERRORS: $RUN_OUTPUT"
        fi
else
        error $COMPILE_OUTPUT
fi

. _breakdown.sh
```

```
/* This is a sample comment. */
```

Unit Test: convoperand

```
#!/bin/bash
TEST_DESC="kernel ** channel"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "kernel ** channel is compiled"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Image img;

Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | @sobelGx | @sobelGy | sobelG;

Image img2 = sobel ** img:Red;
```

Unit Test: cstring1

```
#!/bin/bash
TEST_DESC="Preprocessor in C-String"
. _buildup.sh


compile_it
if [ $ERRORS -eq 0 ]; then
    ERRORS=1
    error "Allowed preprocessor in C-String!"
else
    ERRORS=0
fi

. _breakdown.sh
```

```
Calc c := #[#ifdef CALC]#;
```

Unit Test: cstring2

```
#!/bin/bash
TEST_DESC="Comment in C-String"
. _buildup.sh


compile_it
if [ $ERRORS -eq 0 ]; then
    ERRORS=1
    error "Comment in C-String!"
else
    ERRORS=0
fi

. _breakdown.sh
```

```
Calc c := #[//remove trailing semicolon!]#;
```

Unit Test: cstring3

```
#!/bin/bash
TEST_DESC="Accept empty C-string"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        run_it
        ERRORS=$?
        if [ $ERRORS -ne 0 ]; then
                error "Binary did not exit normally: $ERRORS: $RUN_OUTPUT"
        fi
else
        error $COMPILE_OUTPUT
fi

. _breakdown.sh
```

```
Calc c := #[]#;
```

Unit Test: cstring4

```
#!/bin/bash
TEST_DESC="Unclosed function call in C-String"
. _buildup.sh


compile_it
if [ $ERRORS -eq 0 ]; then
    ERRORS=1
    error "Unclosed function call in C-String!"
else
    ERRORS=0
fi

. _breakdown.sh
```

```
Calc c := #[sqrt(2]#;
```

Unit Test: cstring5

```
#!/bin/bash
TEST_DESC="Unclosed parentheses in C-String"
. _buildup.sh


compile_it
if [ $ERRORS -eq 0 ]; then
    ERRORS=1
    error "Unclosed parentheses in C-String!"
else
    ERRORS=0
fi

. _breakdown.sh
```

```
Calc c := #[2*(3-(4+5)]#; /*mismatched parentheses*/
```

Unit Test: cstring6

```bash
#!/bin/bash
TEST_DESC="Sanity check. C-String should work."
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
	run_it
	ERRORS=$?
	if [ $ERRORS -ne 0 ]; then
		error "Binary did not exit normally: $ERRORS: $RUN_OUTPUT"
	fi
else
	error $COMPILE_OUTPUT
fi

. _breakdown.sh
```

```
Calc c := #[1+2+3]#;
```

Unit Test: defcalc1

```
#!/bin/bash
TEST_DESC="Calc simultaneous declaration and definition" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Calc id<Uint8> := [1 / 1] { 1 0 0 , 0 1 0 , 0 0 1 };
```

Unit Test: defcalc2

```
#!/bin/bash
TEST_DESC="Calc declared but not defined"
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
  error "Calc should not be allowed to be declared without being defined"
else
  ERRORS=0
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Calc id<Uint8>;
```

Unit Test: defcalc3

```
#!/bin/bash
TEST_DESC="Calc defined after it has been declared"
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
  error "Calc should not be allowed to be declared without being defined"
else
  ERRORS=0
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Calc id<Uint8>;

id := [1 / 1] { 1 0 0 , 0 1 0 , 0 0 1 };
```

Unit Test: defchannels

```
#!/bin/bash
TEST_DESC="Uninitialized images don't have default RGB channels"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Accessed Red channel on undefined Image!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Image a;
a:Red; /* should fail */
```

Unit Test: equality-trans

```
#!/bin/bash
TEST_DESC="Transfer equality" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    ERRORS=$?
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image a;
Image b;
Image src = imgread("ucla.png");

a = b = src;

a:Red;
```

Unit Test: id-overlap

```
#!/bin/bash
TEST_DESC="Declare different types with same identifier" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    ERRORS=1
    error "Compiled with overlapping identifier"
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
    ERRORS=0
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image x;
Calc x;
Kernel x;
```

Unit Test: image-defeq

```bash
#!/bin/bash
TEST_DESC="Define-Equals on Image object"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Allowed Define-Equals on Image object!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Image s := imgread("ucla.jpg");
```

Unit Test: image-eq-image

```
#!/bin/bash
TEST_DESC="Image Equals Image"
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    ERRORS=$?
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image s = imgread("ucla.png");
Image t = s;
```

Unit Test: image-oreq-image

```
#!/bin/bash
TEST_DESC="Test or-equal from Image to Image"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Or-equalled image to image!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Image s = imgread("ucla.png");
Image t = imgread("ucla2.png");
t |= s;
```

Unit Test: imgchannel1

```bash
#!/bin/bash
TEST_DESC="Manipulate default RGB channels" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image s = imgread("ucla.png");
Image t;
t:Red = s:Green;
```

Unit Test: imgchannel2

```
#!/bin/bash
TEST_DESC="Append undefined channel" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    ERRORS=1
    error "Added undefined channel to image!"
else
    # compilation errors are here, and you can print them
    # using the "error" function
    ERRORS=0
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image s;
Calc C;
s |= C;
```

Unit Test: imgchannel3

```
#!/bin/bash
TEST_DESC="Append/Assign matrix channel" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    ERRORS=1
    error "Added matrix Calc to Image!"
else
    # compilation errors are here, and you can print them
    # using the "error" function
    ERRORS=0
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image s;
Calc C := [1/1] { 0 0 0 , 0 1 0 , 0 0 0 };
s |= C;
```

Unit Test: imgchannel4

```
#!/bin/bash
TEST_DESC="Add a Calc defined with a C statement" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
        run_it
        ERRORS=$?
        if [ ! $ERRORS -eq 0 ]; then
                error "$RUN_OUTPUT"
        fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
        error "$COMPILE_OUTPUT"
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image s = imgread("ucla.png");
Calc C := #[Red + Blue]#;
s |= C;
```

Unit Test: imgcopy

```
#!/bin/bash
TEST_DESC="Copy an image to a new location."
. _buildup.sh

INPUT_IMG="ucla.png"
OUTPUT_IMG="ucla_out.png"

FAILED=0
# Compile the imgcopy program
compile_it
if [ $ERRORS -eq 0 ]; then
        run_it "$INPUT_IMG" "$OUTPUT_IMG"
        if [ ! $ERRORS -eq 0 ]; then
                error "$RUN_OUTPUT"
        fi
else
        error "$COMPILE_OUTPUT"
fi

# Verify that it produced an output file
if [ $ERRORS -eq 0 ]; then
        if [ ! -f "$OUTPUT_IMG" ]; then
                ERRORS=1
                error "No output image file produced"
        fi
fi

# Verify the file is identical to the input
if [ $ERRORS -eq 0 ]; then
        compare "$INPUT_IMG" "$OUTPUT_IMG"
        ERRORS=$?
        if [ $ERRORS -ne 0 ]; then
                error "Image produced is not identical to original image"
        fi

        rm -f "$OUTPUT_IMG"
fi

. _breakdown.sh
```

```

Image img;
img = imgread("ucla.png");
imgwrite(img, "png", "ucla_out.png");
```

Unit Test: imgread

```
#!/bin/bash
TEST_DESC="imgread function" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image s;
s = imgread("ucla.jpg");
```

Unit Test: imgread-bad

```
#!/bin/bash
TEST_DESC="imgread function: assignment before declaration"
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    ERRORS=1
    error "Compiled with assignment before declaration!"
else
    # compilation errors are here, and you can print them
    # using the "error" function
    ERRORS=0
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
s = imgread("ucla.jpg");
Image s;
```

Unit Test: imgread-bad2

```
#!/bin/bash
TEST_DESC="imgread with two arguments"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Ran imgread with two arguments!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
imgread("ucla.png","ucla2.png");
```

Unit Test: imgread-bad3

```
#!/bin/bash
TEST_DESC="imgread with matrix argument"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Ran imgread with matrix argument!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
imgread({ 0 0 0, 0 0 0, 0 0 0 });
```

Unit Test: imgwrite-bad1

```
#!/bin/bash
TEST_DESC="imgwrite with matrix argument"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Ran imgwrite with matrix argument!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
imgwrite({ 0 0 0, 0 0 0, 0 0 0 });
```

Unit Test: imgwrite-norgb

```
#!/bin/bash
TEST_DESC="imgwrite without RGB channels"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Wrote image without RGB channels"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Image a = imgread("ucla.png");
Calc c := #[1]#;
Calc d := #[2]#;
Kernel k = | c | d;
Image b = a:Red ** k;
imgwrite(b, "png", "fail.png");
```

Unit Test: invalid1

```
#!/bin/bash
TEST_DESC="Test undefined identifiers"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Compiled with an invalid identifier!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
InvalidIdentifier;
```

Unit Test: keyword-id

```
#!/bin/bash
TEST_DESC="Test using Keyword Kernel as an identifier"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "an identifier named Kernel"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Calc Kernel := [1/1]{1 1 1, 1 1 1, 1 1 1};
```

Unit Test: matrix1

```
#!/bin/bash
TEST_DESC="Matrix w/o scale"
. _buildup.sh


compile_it
if [ $ERRORS -eq 0 ]; then
        run_it
        if [ ! $ERRORS -eq 0 ]; then
                error "$RUN_OUTPUT"
        fi
else
        error "$COMPILE_OUTPUT"
fi

. _breakdown.sh
```

```
Calc mat := { 0 0 0 , 0 0 0 , 0 0 0 };
```

Unit Test: matrix2

```
#!/bin/bash
TEST_DESC="Matrix with unequal rows"
. _buildup.sh


compile_it
if [ $ERRORS -eq 0 ]; then
    ERRORS=1
    error "Defined matrix with unequal number of elements per row"
else
    ERRORS=0
fi

. _breakdown.sh
```

```
Calc mat := [1 / 1] { 1 1 1 , 1 1 , 1 1 1 1 };
```

Unit Test: matrix3

```
#!/bin/bash
TEST_DESC="Matrix ending with comma"
. _buildup.sh


compile_it
if [ $ERRORS -eq 0 ]; then
    ERRORS=1
    error "Defined matrix ending with comma!"
else
    ERRORS=0
fi

. _breakdown.sh
```

```
Calc m := { 0 0 0 , 0 0 0 ,;
```

Unit Test: matrix4

```
#!/bin/bash
TEST_DESC="Matrix with multiple right brackets"
. _buildup.sh


compile_it
if [ $ERRORS -eq 0 ]; then
    ERRORS=1
    error "Defined matrix with multiple right brackets!"
else
    ERRORS=0
fi

. _breakdown.sh
```

```
Calc m := { 0 0 0 } 0 0 0 } 0 0 0 };
```

Unit Test: rval-calc

```
#!/bin/bash
TEST_DESC="Unused Calc variable" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Calc c := { 1 0 , 0 1 };
c;
```

Unit Test: rval-chanref

```
#!/bin/bash
TEST_DESC="Unused Channel reference" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image srcimg = imgread("ucla.png");

srcimg:Red;
```

Unit Test: rval-conv

```
#!/bin/bash
TEST_DESC="Unused convolution result" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image srcimg = imgread("ucla.png");

Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
Calc sobelG<Uint8>:= #[sqrt(sobelGx*sobelGx + sobelGy*sobelGy)]#;

srcimg |= Lum;

Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | @sobelGx | @sobelGy | sobelG;

srcimg:Lum ** sobel;
```

Unit Test: rval-cstr

```
#!/bin/bash
TEST_DESC="Unassigned C-String" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
#[Red + Green + Blue]#;
```

Unit Test: rval-image

```
#!/bin/bash
TEST_DESC="Unused image variable" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Image s = imgread("ucla.jpg");
s;
```

Unit Test: rval-imgread

```bash
#!/bin/bash
TEST_DESC="Unassigned imgread" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
imgread("ucla.jpg");
```

Unit Test: rval-kernel

```
#!/bin/bash
TEST_DESC="Unused Kernel variable" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | sobelGx | sobelGy;

sobel;
```

Unit Test: rval-matrix

```
#!/bin/bash
TEST_DESC="Unassigned matrix" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
{ 0 1 2 , 3 4 5 , 6 7 8 };
```

Unit Test: sizediff

```
#!/bin/bash
TEST_DESC="Assign channels of different sizes"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        run_it
        if [ $ERRORS -eq 0 ]; then
                ERRORS=1
                error "Assigned channels of different sizes!"
        else
                # The running of the program should fail.
                ERRORS=0
        fi
else
        error "Why didn't this compile?!"
fi

. _breakdown.sh
```

```
Image a = imgread("flatiron.jpg");
Image b = imgread("ucla.png");
a:Red = b:Red;  /* Should fail because of size difference*/
```

Unit Test: sobel

```
#!/bin/bash
TEST_DESC="Compile and test the Sobel filter"
. _buildup.sh

INPUT_IMG="ucla.png"
OUTPUT_IMG="ucla_sobel.png"

compile_it
if [ $ERRORS -eq 0 ]; then
        run_it "$INPUT_IMG" "$OUTPUT_IMG"
        if [ ! $ERRORS -eq 0 ]; then
                error "$RUN_OUTPUT"
        fi
else
        error "$COMPILE_OUTPUT"
fi

if [ $ERRORS -eq 0 ]; then
        EXPECTED="ucla_sobel_expected.png"
        compare "$OUTPUT_IMG" "$EXPECTED"
        ERRORS=$?
        if [ ! $ERRORS -eq 0 ]; then
                error "Output image $OUTPUT_IMG  does not match $EXPECTED"
        fi
        rm "$OUTPUT_IMG"
fi

. _breakdown.sh
```

```
Image srcimg = imgread(1);

Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
Calc sobelG<Uint8>:= #[sqrt((float)sobelGx*sobelGx + (float)sobelGy*
    sobelGy)]#;
Calc sobelTheta<Angle>:= #[atan((float)sobelGy/(float)sobelGx)]#;

srcimg |= Lum;

Calc sobelGx<Uint8> := [1 / 1]{ -1  0 +1 , -2  0 +2 , -1  0 +1 };
Calc sobelGy<Uint8>:=[1 / 1]{ +1 +2 +1 ,  0  0  0 , -1 -2 -1 };

Kernel sobel = | @sobelGx | @sobelGy | sobelG;
sobel |= sobelTheta;

Image edges = srcimg:Lum ** sobel;
Image output;
output:Red   = edges:sobelG;
output:Green = edges:sobelG;
output:Blue  = edges:sobelG;

imgwrite( output, "png", 2);
```

Unit Test: string1

```
#!/bin/bash
TEST_DESC="Consecutive String Constants"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        run_it
else
    error "Couldn't compile consecutive string constants, e.g. \"foo\" \"bar\""
fi

. _breakdown.sh
```

```
Image a = imgread("foo" "bar" ".jpg");
```

Unit Test: undefined1

```
#!/bin/bash
TEST_DESC="Test undefined identifiers"
. _buildup.sh

compile_it
if [ $ERRORS -eq 0 ]; then
        ERRORS=1
        error "Compiled with an undefined channel!"
else
        # we purposely succeed b/c this test was
        # designed to fail!
        ERRORS=0
fi

. _breakdown.sh
```

```
Image srcimg = imgread("ucla.png");

Calc Lum := #[(3*Red + 6*Green + 1*Blue)/10]#;
srcimg |= Lum;

Image output;
output:foo = srcimg:foo; /* srcimg:foo doesn't exist! */
```

Unit Test: zerocalc1

```
#!/bin/bash
TEST_DESC="Calc definition with zero multiplier" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    run_it
    # STDOUT/STDERR are put into "$RUN_OUTPUT"
    if [ ! $ERRORS -eq 0 ]; then
        error "Problem running this test!"
    fi
else
    # compilation errors are here, and you can print them
    # using the "error" function
    error $COMPILER_OUTPUT
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Calc id<Uint8> := [0 / 1] { 1 0 0 , 0 1 0 , 0 0 1 };
```

Unit Test: zerocalc2

```
#!/bin/bash
TEST_DESC="Calc definition with divide by zero" #change this
. _buildup.sh

# This will compile the test CLAM source file
# setting ERRORS to the compiler's return value
compile_it

if [ $ERRORS -eq 0 ]; then
    # This runs the compiled output, accepts
    # any command-line arguments, and sets ERRORS
    # to the return value of the new binary
    ERRORS=1
    error "Allowed divide by zero!"
else
    # compilation errors are here, and you can print them
    # using the "error" function
    ERRORS=0
fi

# At the end of the test, set ERRORS=0 if
# there were no errors. Nonzero means fail.
# ERRORS=1
# error "This test auto-fails"

. _breakdown.sh
```

```
Calc id<Uint8> := [1 / 0] { 1 0 0 , 0 1 0 , 0 0 1 };
```

Appendix B

Project Version Control History

The CLAM project used `git` [5] as its version control system. Here we provide the output of the `git-shortlog` program with *Merge* commits filtered out, followed by a more complete revision control history.

B.1 Project Repository `git` 'shortlog'

```
Jeremy C. Andrus <jeremya@cs.columbia.edu> (138):
      Initial import
      initial latex document import
      renamed and moved sample code file
      updated source example
      added gitignore file
      Updated LaTeX source - edit proposal.tex for proposal (type make to build)
      removed old code sample
      removed unnecessary file
      updated gitignore
      removed text from intro
      updated formatting: changed name to CLAM
      Initial language proposal document
      nits
      initial import of calculator example
      default to LRM compilation: edit manual.tex
      cleanall -> distclean
      updated gitignore
      Major framework update
      beautify tester + add compile output checking
      no binaries in the git repo please
      check for clam binary before doing regression testing
      updated example to follow better formulated grammar
      initial LRM import: will probably break this up into files for easier collaboration
      update gitignore
      split LRM into a couple of files, more basic type work
      split LRM into multiple files
      nits and bugfixes to lexigraphical conventions
      more nits/updates to lex...
      update expression and object discussion (stil incomplete)
      update code listing
      more expression and object additions
      remove the word annoying
      Major initial code dump!
      Massive commit which adds Syntax error handling
      added some basic test definitions (no results)
      Full string/escaped-C/Matrix-definition parsing
      another test (basically mixed-up sobel)
      count newlines in comments and strings
      Added a verifier stub
      don't clear link-order file on every build!
      Added the basic verifier framework
      Added option to output generated C code
      more verifier work... things are in flux :-)
```

```
Major work on the verifier
Tweaks to test framework: give example of invalid compile test
Initiali import of stb library for imgread/imgwrite
Added imgread/imgwrite interface functions
filled in the clam_img_alloc function
Compile image library, use it in final link
adding gitignore
Unit test documentation update + 1 new simple test
Add debugging info to Ocaml binary
Nearly complete verifier!
removing unnecessary file
remove Clamtypes reference
Fixed imgwrite error in PNG output
A whole bunch of C machinery to support the backend
C template for use in creating the backend
Remove ChanEval, Add basic escaped-C parsing
ImageT() -> CalcT in '=' error message
Replaced tab characters with spaces... sorry, I'm anal like that
More OCaml-like equal-row verification
Add default channel names to an image
Checking for ImageT |= CalcT(matrix)
Print error message from compile_it
Disallow channel assignment from uninitialized CalcT
Merge remote-tracking branch 'origin/semantic' into test
Small bugs in post-merge code
reverse the output of the verifier
remove debugging
Compress translation
Updated C-backend template: still missing convolution imp...
A little more detail on C-style convolution backend
Updated C backend structure
Convolution works in the C-backend example.
Convolution operator restrited to ChanRef * Id
Auto C backend header generation
Fixed backend bugs in channel assignment
Revert "Changed _buildup.sh (temporarily) so that failure for gcc to compile doesn't count as error
    "
Convolution prep: functionalize
Convolution function generation works
Updated sobel example to be correct
Added support for C-style casts in escaped-c strings
Partial kernel assignment
Sobel compiles
Sobel calculation works!
Append channels to the image properly
Added expected sobel output
Fixed multiply defined variables
Fixed scanner bugs + added default R/G/B calc types
Passes the 1 calc kernel test
more testing stuff
Updated gitignore
More formal integration of generated files
Default to Final Report generation + fix Latex document build
Starting with Edward's suggested sub-points in each chapter
Renamed parse_util.ml to parseutil.ml (to be nicer to Latex file include)
Updated formatting for code listing
Initial Code Listing Appendix (still missing a few things)
Structure for individual 'Lessons Learned' sections
renamed stb_image_* files to be nice to stupid Latex...
renamed some more underscore files... this gets old REALLY fast...
Automatically generate a git-shortlog during "make"
Added all unit tests to source listing (+src listing cleanup)
Fixed single-calc assignment bug + Code Cleanup (for report listing)
Updated sobel source to reflect new Kernel syntax
nits/updates in tutorial
Moved Git history to separate appendix
nits in Intro and Tutorial
Include a full (colorized) git history
remove last references to CHANNELT
added a label in here that I needed elsewhere
added a label here which I needed elsewhere
Added a Sobel reference
Overview of "Project Plan" - other sections coming shortly
Fixed figure references and minor formatting
Minor tweak to the matrix definition
added a label that I needed elsewhere
Remove old Sobel example... new one coming soon
Added labels and imported a couple more useful packages
Silly/Anal name fix
Filled in the Project Plan section completely
Added the ability to concatenate string constants
```

```
      Cleanup in the LRM + nits elsewhere
      Lessons learned - jeremy
      Format adjustments for code listings
      Fixed RGB channel assignment in output image.
      removed extraneous text
      FORMAT HACK
      Added better/more realistic examples (complete with images)
      nits
      BUGFIX: textt{ -> texttt{
      Updated sizediff tests: there is no way to test this at compile time
      Removed cruft, renamed paper.tex -> clam.tex
      Added a README file
      Added section references and the string1 test
      listing syntax highlights -> CLAM, Added string1 test
      Added the presentation PDF to the doc/ dir


Robert Martin <rdm2128@columbia.edu> (83):
      Add some hypothetical sample code
      Add language basic language syntax concepts and simple example problem to the proposal
      Add removal of 'clam' binaries to 'make clean'
      Simple as possible program that hooks C functions into ocaml, in case we need to do this
      Modify backend to dynamically write C code for the calculator instead of doing calculations at
          compile time
      Begin work on LRM...
      Update reserved keyword list
      Add statements text and fix some typos
      Update some more misc. sections
      Merge
      Work on escaped C expressions, fix keywords, and talk about convolutions
      Redo the test harness, write some simple unit tests
      Improve the README for the unit tests
      Possible to print AST by adding the "-t" switch. The AST printing still needs some polishing.
      Add documentation on how we will accept command-line strings into our langauge for imgread and
          imgwrite
      Split clamtypes into two logical parts: types and environment. Add environ.ml to hold environment
          code.
      Move printing functions to printer.ml. Update other code so it uses the printer.ml stuff.
      Fix formatting bug in --help message
      Refactor: create a 'string_of_libf' to translate ImgRead and ImgWrite into strings
      Change naming convention for output binaries from *.clam to *.out (because *.clam are the source
          files); Add skeleton for C generation.
      Fix error in syntax of imgcopy test, minor cosmetic fix to AST printing, comment-out unused code in
           backend (will be deleted soon)
      Add skeletal code for backend C generation
      Remove the "-fast" flag so CLAM compiles
      Remove clamtypes, move the types to their associated ml files instead.
      Beautify the sobel source and fix typo in image name
      Improve backend structure, add some stubs where code will need to be generated
      Fix bug where declarations were in the wrong order
      Additional structure for backend.ml, code now makes printfs instead of actually doing anything....
      Replace JPG with PNG, because our image library does not support JPG
      Add structure for verified syntax tree (vast.mli)
      Change structure of environment variables, tweak vast, change how printer works
      Create skeleton for verifier and backend to create a nice VAST instead of the AST
      Verifier can recognize some declarations and backend can start to recognize verified AST
      Clam compiles. Sobel parses but is not being very strongly verified. Everything that gets generated
           can be understood by backend
      Verifier starting to work, but it currently thinks every expression is an ImageT.
      Manual Merge branch 'master' of git.ncl.cs.columbia.edu:phd/jeremya/plt.f11
      Add "Unknown" atomic type for some Calc's. Switched so every expression parses as a Calc. Still
          doesn't verify anything
      Verifier has a very limited set of things it can verify
      Added ability to parse Channel References
      Big improvement to producing a much nicer AST
      Verified Environment can detect lvalues and rvalues
      Allow environment to affected by ChanRefs
      Move verifier.ml to semantic.ml. Getting ready to pull verifier
      Further prepare for the merge. Things seems to work at this point
      Remove environ.ml and verifier.ml to try and help merge
      Unify things so they're pretty. We pass 13 tests.
      Change tests because this branch doesn't allow "Calc" to be declared without begin defined
      Improve matrix semantic checking
      Only allow CStrings to be OrEq with images or kernels
      Fix bug where only CString was accepted as an R-Value for |= operator
      Modify Sast stuff to use the Environment built in the verifier.
      Prune some extra code from Semantic
      Change all "Failure" exceptions in Semantic to "SemanticFailure" exceptions
      Translate matrices into 'int list list's rather than 'BInt list list's
      Add some basic structure to the backend to handle the Sast
      Disallow VExpr from being a filename or an image format
      Minor cleaning-up in backend.ml
      Begin traversing the Sast tree in backend
      Further improve backend and semantic Sast
      Finish (I think) the conversion from Ast -> Sast. Backend has some stubs.
      Further streamline code in Semantic, add more hooks in backend.
      Handle Image types internally as Enum instead of strings
      ImgWrite, ChanRef handled by backend
      Add more code-writing ability to backend
      ImgRead added to backend
      Start calc assignment, add "matrix" to environment
      Add backend checking to distinguish between image 'assignments' and image 'copies'
      Add kernel_copy to backend, fix bug in Makefile, add Null check to kernel_copy
      Add a couple more stubs to clam.h and clam.c, slightly improve back-end code, fix a small bug in
          clam.h
      Add clam_clib.ml to PHONY so it always rebuilds
      Improve Verifier handling of ImgRead and ImgWrite
      Add # of argument runtime check into a.out.
      Add ability to change a sMatrix -> the { { } } C format
      Fix some minor bugs in clam.h
      Add backend for Calc Matrix -> Generated C code
      Move definition of bail to clam.h
      Fix ordering of operands to g++ in clamsys
      Update the introduction to more accurately reflect Clam as it currently exists
      Insert text for design/architecture section, minus any diagrams
      Add some diagrams and tables to the "Design" section
      Fix wrapping on diagram
      Fix problem with table of tokens
      Add lessons-learned section


Kevin Sun <kfs2110@columbia.edu> (67):
      Added a basic shell script for testing, and some examples
      Added some tests for imgread - must declare before assignment
      Added some tests for definition of Calc
      Renamed defcalc.* to defcalc1.*
      Added tests for multiplier [0 / 1] and [1 / 0]
      Added tests for overlapping identifiers, and channel manipulation
      Changed verifier to catch divide-by-zero error in matrix definition.     Is this right?
      Added tests for #[]# operators, and more channel manipulation stuff
      Corrected "cstrdef.test"
      Clarified description of "imgchannel1.test"
      Modified imgchannel2.test
      Added more tests for adding Calcs to Images
      Added matrix-related tests:     matrix1.test: Matrices without scaling factors are not accepted.
              matrix2.test: Matrices with unequal row length are accepted.???
      Edited parser to accept matrices without scalers in front.     ( [1 / 1] is the default )
      Added tests for image with various assignment operators (:=, =, |=)
      Changed tests to use assignment operators properly
      Modified verifier to check that matrix rows are of equal length
      Removed reference to $() in LRM, since it's no longer needed
      Outline for tutorial.     Copied intro from proposal into Intro for final report.
      Test for @ operator
      Added "r-value" tests: variable and operations work even if return values are ignored
      Test failed: undefined images still have default channels
      Test failed: assign channels of different sizes
      Added two other images for testing
      Changed/added tests for image |= image (doesn't work, or distinguish size)
      Test succeeded: Equality is transferred (Image a = b = imgread(...))
      Test failed: wrote image without valid RGB channels
      Tests failed: imgread accepts multiple arguments, and invalid argument types
      Test failed: imgwrite ran with bad arguments
      Changed imgwrite_norgb to write a file with only R and B
      Tests failed: Accepted ill-formed matrices.     I'll fix this now.
      Fixed matrix definition in parser to disallow bizarre definitions like:     { 0 0 0 } 1 1 1 , 2 2 2
           } 3 3 3 ,
      Tests passed: disallow comments or preprocessor in C-String calc definition
      (Backend seems to be in flux, will check back on this later.)
      Added checks for cstring validity.     Getting some "C-Backend" errors at the moment.
      Add more C-string tests: open parentheses test and "sanity check" test that should pass.     Both
          throw Backend errors right now. (which means the open parentheses isn't caught earlier)
      Changed _buildup.sh (temporarily) so that failure for gcc to compile doesn't count as error     (
          Otherwise gets lots of false positives/negatives)
      Temporarily changed tests/_buildup.sh to ignore GCC errors
      Changed keyword_id.clam to actually be meaningful.     ("Red" technically isn't a keyword; "Kernel"
           certainly is.)
      Changed scanner to actually catch mismatched parens/calls in escaped C     (Wasn't maintaining
          level correctly before)     Two more tests pass now
      Oops sorry was toying with _buildup.sh again, reverted now.
      Reversed result of cstring3.test so accepting empty C-strings is good
      Changed imgchannel2.clam to match description
      Fixed imgchannel4.test
      Fixed equality_trans.test to accept
      Fixed image_eq_image.test to accept
      Disallow " and ' in escaped C
      Copied Tutorial text from presentation slides.     To do: Fix formatting (headers/syntax coloring),
              and reorganize text to make more sense in context.
      Basic formatting for tutorial, still have to reorganize text.     Put in some "lessons learned".
      Finished tutorial. Commented and reformatted sobel.clam (in docs/src/).
      Edited lrm-statements and lrm-types     Changed lrm-statements to mention "useless statements",
             such as "imgread(1);" with no assignment.     Changed lrm-types to mention defualt atom-
          values,     removed section on type qualifiers since we don't actually have any anymore.
      Removed description of non-existent "||" operator.     Changed description of "|" to apply only to
          Calcs, and account for modified syntax
      Refined assignment operator descriptions to clarify type constraints.     Removed section on non-
          existent "^=" operator.
      Reworded "Statements", split "Calculation Constants" into     "Matrix Constants" and "Escaped C
          Constants",     removed part about 'a' and 'f' for Angles and Floats
      Undid the splitting of "Calculation Constants".     Realized it was partially redundant with "
          Expressions".
      Changed Kernel description in "Objects", because Kernels have to     be defined using Calc *
          identifiers* and not just calc constants/expressions.
      Removed references to Float and Channel (as a type)
      Did "Expressions" section of "Grammar", based off of parser code
      Finished Grammar portion of LRM
      Minor details in Grammar
      Added section headers for test plan.
      More test plan work
      Summary paragraphs for each section in test plan.
      Deleted now-irrelevant image-oreq-image2.clam and removed reference from appendix.     Removed
          pfggantt package from paper.tex since it is not used and was causing compile errors.
      More Test Plan
      More test plan. Removed "addker2img" test because it's redundant     and checks multiple features
          at once.
      Test Plan done and itemized. Reinserted \usepackage{pgfgantt} to paper.tex


Yongxu Zhang <yz2419@columbia.edu> (11):
      Edited intro.tex
      functions about covolution
      variables could be assigned with shorter uint
      an image could be assigned with a smaller matrix
      correct the  descriptions of test files
      An Eq, besides DefEq, can define a Calc
      Added type checking for assignment operators with Calc
      Calc should not be assigned by another Calc, so remove typeconvertion.test
      some unit tests discussed in the morning
      test plan, any suggestions?
      lessons learned
```

B.2 Full `git` Log

The following log was generated using the command:

```
git log --color --stat --no-merges --pretty=format:"%h: %Cblue%aN <%aE>%Creset%nDate: %aD%nSubject: %s%
    nContent: %b"
```

```
dbc1e3e: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 21:19:07 -0500
Subject: Added the presentation PDF to the doc/ dir
Content:
 doc/clam-presentation.pdf |  Bin 0 -> 264761 bytes
 1 files changed, 0 insertions(+), 0 deletions(-)

6cd70bf: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 21:14:02 -0500
Subject: listing syntax highlights -> CLAM, Added string1 test
Content:
 doc/appendix.tex |  109 +++++++++++++++++++++++++++--------------------------
 1 files changed, 56 insertions(+), 53 deletions(-)

d57e907: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 21:13:48 -0500
Subject: Added section references and the string1 test
Content:
 doc/testing.tex |   17 ++++++++++++-----
 1 files changed, 12 insertions(+), 5 deletions(-)

bee8acc: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 20:55:25 -0500
Subject: Added a README file
Content:
 README |   39 +++++++++++++++++++++++++++++++++++++++
 1 files changed, 39 insertions(+), 0 deletions(-)

c10f54b: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 20:52:41 -0500
Subject: Removed cruft, renamed paper.tex -> clam.tex
Content:
 .gitignore              |   12 +++---
 chook/Makefile          |   17 -------
 chook/dummy.mli         |    3 -
 chook/test.ml           |    7 ---
 chook/wrap.c            |   22 ---------
 doc/Makefile            |    2 +-
 doc/clam.bib            |   38 ++++++++++++++++
 doc/clam.tex            |  113 +++++++++++++++++++++++++++++++++++++++++++++++
 doc/clam.tex.latexmain  |    1 +
 doc/paper.bib           |   38 ----------------
 doc/paper.tex           |  113 -----------------------------------------------
 doc/paper.tex.latexmain |    1 -
 12 files changed, 159 insertions(+), 208 deletions(-)

12b0c5e: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 20:38:57 -0500
Subject: Updated sizediff tests: there is no way to test this at compile time
Content:
 clam/tests/sizediff.test |   14 +++++++++-----
 1 files changed, 9 insertions(+), 5 deletions(-)

2ea56ea: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 14:57:54 -0500
Subject: BUGFIX: textt{ -> texttt{
Content: what version of TeX are you using that this actually compiles?!
please check before pushing :-)

 doc/testing.tex |   24 ++++++++++++------------
 1 files changed, 12 insertions(+), 12 deletions(-)

6dd9546: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 22 Dec 2011 14:45:37 -0500
Subject: Test Plan done and itemized. Reinserted \usepackage{pgfgantt} to paper.tex
Content:
 clam/tests/ckernel.clam |    3 +-
 doc/paper.tex           |    1 +
 doc/testing.tex         |  131 ++++++++++++++++++++++++++++++++---------------
 3 files changed, 92 insertions(+), 43 deletions(-)

658b9b3: Kevin Sun <kfs2110@columbia.edu>
```

```
Date: Thu, 22 Dec 2011 14:10:25 -0500
Subject: More test plan. Removed "addker2img" test because it's redundant and checks multiple features at
     once.
Content:
 clam/tests/addker2img.clam |   11 -----------
 clam/tests/addker2img.test |   15 ---------------
 doc/appendix.tex           |    3 ---
 doc/testing.tex            |   31 +++++++++++++++++++------------------
 4 files changed, 19 insertions(+), 41 deletions(-)

9c422c5: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 22 Dec 2011 13:56:11 -0500
Subject: More Test Plan
Content:
 clam/tests/undefined1.clam |    6 +--
 doc/testing.tex            |  154 +++++++++++++++++++++++++++++++++-------------------
 2 files changed, 109 insertions(+), 51 deletions(-)

b8ce776: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 22 Dec 2011 12:46:50 -0500
Subject: Deleted now-irrelevant image-oreq-image2.clam and removed reference from appendix. Removed
    pfggantt package from paper.tex since it is not used and was causing compile errors.
Content:
 doc/appendix.tex |    3 ---
 doc/paper.tex    |    1 -
 doc/testing.tex  |    6 +++---
 3 files changed, 3 insertions(+), 7 deletions(-)

47a2674: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 22 Dec 2011 12:37:06 -0500
Subject: Summary paragraphs for each section in test plan.
Content:
 doc/testing.tex |   13 +++++++++++++
 1 files changed, 13 insertions(+), 0 deletions(-)

0b562fc: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 22 Dec 2011 12:25:20 -0500
Subject: More test plan work
Content:
 clam/tests/image-oreq-image.clam  |    2 +-
 clam/tests/image-oreq-image.test  |   26 ++++--------------
 clam/tests/image-oreq-image2.clam |    3 --
 clam/tests/image-oreq-image2.test |   15 ----------
 doc/testing.tex                   |   55 ++++++++++++++++++++++++++++++++++++---
 5 files changed, 58 insertions(+), 43 deletions(-)

d1a7b59: Yongxu Zhang <yz2419@columbia.edu>
Date: Thu, 22 Dec 2011 11:44:45 -0500
Subject: lessons learned
Content:
 doc/lessons-yongxu.tex |    3 +++
 1 files changed, 3 insertions(+), 0 deletions(-)

8cbdf12: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 22 Dec 2011 11:35:09 -0500
Subject: Added section headers for test plan.
Content:
 doc/testing.tex |   17 ++++++++++++++++-
 1 files changed, 16 insertions(+), 1 deletions(-)

9778fe7: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 04:35:27 -0500
Subject: nits
Content:
 doc/manual.tex |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

7b15dcf: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 04:29:58 -0500
Subject: Added better/more realistic examples (complete with images)
Content:
 clam/tests/clamsrc/blur.clam    |    4 --
 clam/tests/clamsrc/segment.clam |   18 ++++++++++++
 doc/figures/lena-blur.png       |  Bin 0 -> 357031 bytes
 doc/figures/lena-seg.png        |  Bin 0 -> 26523 bytes
 doc/figures/lena.png            |  Bin 0 -> 527956 bytes
 doc/manual.tex                  |   60 +++++++++++++++++++++++++++++++++++++++++--
 doc/paper.tex                   |    2 +-
 doc/src/blur.clam               |   19 +++++++++++++
 doc/src/segment.clam            |   18 ++++++++++++
 9 files changed, 113 insertions(+), 8 deletions(-)
```

```
61e075c: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 04:29:38 -0500
Subject: FORMAT HACK
Content:
 doc/plan.tex |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

bdb3ecd: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 04:28:21 -0500
Subject: removed extraneous text
Content:
 doc/lessons.tex |    3 ---
 1 files changed, 0 insertions(+), 3 deletions(-)

3077386: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 03:42:41 -0500
Subject: Fixed RGB channel assignment in output image.
Content: The Red Green and Blue image channels are now explicitly searched for in
imgwrite() and there is no longer a dependance on the order of
assignment :-)

 clam/libstb/clam.c             |   62 +++---------------------------------------
 clam/libstb/clam.h             |   71 ++++++++++++++++++++++++++++++++++++++++-
 clam/libstb/stb-image-write.c |   11 ++-----
 3 files changed, 78 insertions(+), 66 deletions(-)

f62cddb: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 22 Dec 2011 03:33:14 -0500
Subject: Add lessons-learned section
Content:
 doc/lessons-robert.tex |   11 +++++++++++
 1 files changed, 11 insertions(+), 0 deletions(-)

ab2b671: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 03:24:25 -0500
Subject: Format adjustments for code listings
Content:
 clam/printer.ml  |   31 ++++++++++++-------------------
 clam/sast.mli    |    6 ------
 clam/scanner.mll |    2 +-
 clam/semantic.ml |   40 ++++++++++++++++++++++++----------------
 clam/verifier.ml |    5 ++---
 5 files changed, 39 insertions(+), 45 deletions(-)

17a8f7a: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 03:23:30 -0500
Subject: Lessons learned - jeremy
Content:
 doc/lessons-jeremy.tex |   17 +++++++++++++++++
 1 files changed, 17 insertions(+), 0 deletions(-)

371a547: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 22 Dec 2011 03:05:36 -0500
Subject: Fix problem with table of tokens
Content:
 doc/design.tex |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

edc2d38: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 02:51:28 -0500
Subject: Cleanup in the LRM + nits elsewhere
Content:
 doc/design.tex          |    4 +-
 doc/lrm-expressions.tex |   81 +++++++++++++++++++++++++++++++-----------------
 doc/lrm-lexical.tex     |   20 +++++++++---
 doc/lrm-misc.tex        |   12 ++-----
 doc/lrm-objects.tex     |   16 +++++++--
 doc/lrm-statements.tex  |    3 +-
 doc/lrm-types.tex       |   11 +++---
 doc/manual.tex          |    9 ++++-
 doc/tutorial.tex        |    1 +
 9 files changed, 101 insertions(+), 56 deletions(-)

0b3cc7d: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 01:40:10 -0500
Subject: Added the ability to concatenate string constants
Content:
 clam/scanner.mll        |    4 +++-
 clam/tests/string1.clam |    1 +
 clam/tests/string1.test |   12 ++++++++++++
 3 files changed, 16 insertions(+), 1 deletions(-)

798cf3d: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 01:19:50 -0500
Subject: Filled in the Project Plan section completely
Content:
 doc/plan.tex |   87 +++++++++++++++++++++++++++++++++++++++++++++++++++++++----
 1 files changed, 80 insertions(+), 7 deletions(-)

89b643b: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 01:19:47 -0500
Subject: Silly/Anal name fix
Content:
 doc/design.tex |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

8364fb5: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 01:19:12 -0500
Subject: Added labels and imported a couple more useful packages
Content:
 doc/appendix.tex |    4 +++-
 doc/intro.tex    |    1 +
 doc/lessons.tex  |    1 +
 doc/paper.tex    |    5 ++++-
 doc/tutorial.tex |    1 +
 5 files changed, 10 insertions(+), 2 deletions(-)

1fd0af5: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 22 Dec 2011 00:16:46 -0500
Subject: Remove old Sobel example... new one coming soon
Content:
 doc/manual.tex     |    2 +-
 doc/src/sobel.imp |   31 -------------------------------
 2 files changed, 1 insertions(+), 32 deletions(-)

7276c15: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 21:13:47 -0500
Subject: Minor details in Grammar
Content:
 doc/lrm-grammar.tex |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

5a0a88b: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 21:07:26 -0500
Subject: added a label that I needed elsewhere
Content:
 doc/testing.tex |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

c23d377: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 21:07:06 -0500
Subject: Minor tweak to the matrix definition
Content:
 doc/lrm-grammar.tex |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

6572d1f: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 20:53:23 -0500
Subject: Fixed figure references and minor formatting
Content:
 doc/design.tex |   19 +++++++++----------
 1 files changed, 9 insertions(+), 10 deletions(-)

f5dabe7: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 20:45:13 -0500
Subject: Finished Grammar portion of LRM
Content:
 doc/lrm-grammar.tex |   37 ++++++++++++++++++++++++++++++++++++-
 1 files changed, 36 insertions(+), 1 deletions(-)

f646f01: Robert Martin <rdm2128@columbia.edu>
Date: Wed, 21 Dec 2011 20:22:04 -0500
Subject: Fix wrapping on diagram
Content:
 doc/design.tex |   11 ++++++-----
 doc/paper.tex  |    1 +
 2 files changed, 7 insertions(+), 5 deletions(-)

2b4f9fa: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 20:21:55 -0500
Subject: Did "Expressions" section of "Grammar", based off of parser code
```

```
Content:
 doc/lrm-grammar.tex |   40 +++++++++++++++++++++++++++++++++++++++-
 doc/lrm-objects.tex |    2 +-
 2 files changed, 40 insertions(+), 2 deletions(-)

acc48bf: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 20:17:55 -0500
Subject: Overview of "Project Plan" - other sections coming shortly
Content:
 doc/plan.tex |   40 +++++++++++++++++++++++++++++++++++++++-
 1 files changed, 39 insertions(+), 1 deletions(-)

0b1352c: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 20:17:43 -0500
Subject: Added a Sobel reference
Content:
 doc/paper.bib |    5 +++++
 1 files changed, 5 insertions(+), 0 deletions(-)

792d3e6: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 20:17:23 -0500
Subject: added a label here which I needed elsewhere
Content:
 doc/design.tex |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

e033672: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 20:16:48 -0500
Subject: added a label in here that I needed elsewhere
Content:
 doc/tutorial.tex |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

b336b4b: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 20:16:00 -0500
Subject: remove last references to CHANNELT
Content:
 clam/parser.mly  |    2 +-
 clam/scanner.mll |    1 -
 2 files changed, 1 insertions(+), 2 deletions(-)

25f61ef: Robert Martin <rdm2128@columbia.edu>
Date: Wed, 21 Dec 2011 20:14:56 -0500
Subject: Add some diagrams and tables to the "Design" section
Content:
 doc/design.tex          |   90 +++++++++++++++++++++++++++++++++++++++++------
 doc/figures/layers.png  |  Bin 0 -> 337946 bytes
 doc/figures/layers.xcf  |  Bin 0 -> 564941 bytes
 3 files changed, 78 insertions(+), 12 deletions(-)

ab019e3: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 19:25:01 -0500
Subject: Removed references to Float and Channel (as a type)
Content:
 doc/lrm-expressions.tex |   23 +++++++++++------------
 doc/lrm-lexical.tex     |    6 +-----
 doc/lrm-objects.tex     |   12 ++++++------
 doc/lrm-types.tex       |    4 +---
 4 files changed, 19 insertions(+), 26 deletions(-)

7e73fec: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 18:47:10 -0500
Subject: Changed Kernel description in "Objects", because Kernels have to be defined using Calc *
    identifiers* and not just calc constants/expressions.
Content:
 doc/lrm-expressions.tex |    3 ++-
 doc/lrm-objects.tex     |    7 ++++---
 2 files changed, 6 insertions(+), 4 deletions(-)

546bc21: Robert Martin <rdm2128@columbia.edu>
Date: Wed, 21 Dec 2011 18:46:59 -0500
Subject: Insert text for design/architecture section, minus any diagrams
Content:
 doc/design.tex |   55 ++++++++++++++++++++++++++++++++++++++++++++++++++++---
 1 files changed, 52 insertions(+), 3 deletions(-)

b487603: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 18:42:44 -0500
Subject: Undid the splitting of "Calculation Constants". Realized it was partially redundant with "
    Expressions".
Content:
 doc/lrm-expressions.tex |    3 ++-
 doc/lrm-lexical.tex     |   19 ++-----------------
 doc/lrm-misc.tex        |   12 +++++++-----
 doc/lrm-objects.tex     |    2 +-
 4 files changed, 12 insertions(+), 24 deletions(-)

2c63bb8: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 18:32:45 -0500
Subject: Reworded "Statements", split "Calculation Constants" into "Matrix Constants" and "Escaped C
     Constants", removed part about 'a' and 'f' for Angles and Floats
Content:
 doc/lrm-lexical.tex    |   31 ++++++++++++++++++++++++-------
 doc/lrm-statements.tex |    4 ++--
 doc/lrm-types.tex      |    3 ++-
 3 files changed, 28 insertions(+), 10 deletions(-)

b29fb25: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 18:04:19 -0500
Subject: Refined assignment operator descriptions to clarify type constraints. Removed section on non-
     existent "^=" operator.
Content:
 doc/lrm-expressions.tex |   19 ++++++-------------
 1 files changed, 6 insertions(+), 13 deletions(-)

a948064: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 17:55:35 -0500
Subject: Removed description of non-existent "||" operator. Changed description of "|" to apply only to
     Calcs, and account for modified syntax
Content:
 doc/lrm-expressions.tex |   33 +++++++++------------------------
 1 files changed, 9 insertions(+), 24 deletions(-)

e3ad8c7: Kevin Sun <kfs2110@columbia.edu>
Date: Wed, 21 Dec 2011 17:34:13 -0500
Subject: Edited lrm-statements and lrm-types Changed lrm-statements to mention "useless statements", such
      as "imgread(1);" with no assignment. Changed lrm-types to mention defualt atom-values, removed
     section on type qualifiers since we don't actually have any anymore.
Content:
 doc/lrm-statements.tex |    7 +++++--
 doc/lrm-types.tex      |   14 +++++---------
 2 files changed, 10 insertions(+), 11 deletions(-)

0f37a19: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 17:19:07 -0500
Subject: Include a full (colorized) git history
Content:
 doc/appendix.tex       |    6 ++--
 doc/gen_gitshortlog.pl |   53 ++++++++++++++++++++++++++++++++++++++++++++++++-
 doc/paper.tex          |    2 +-
 doc/plan.tex           |   11 ++++++---
 4 files changed, 62 insertions(+), 10 deletions(-)

d351209: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 12:23:46 -0500
Subject: nits in Intro and Tutorial
Content: Mostly unifying some lexical conventions - specifically, the only words
which we put in texttt{} should be data types or operators. This means
that words like Channel or CString should be put in emph{} the _first_
time they're used, but subsequent instances should be plain text
(capitalization can be maintained)

 doc/intro.tex    |   21 ++++++++++----------
 doc/tutorial.tex |   35 ++++++++++++++++++-----------------
 2 files changed, 29 insertions(+), 27 deletions(-)

dd8a359: Yongxu Zhang <yz2419@columbia.edu>
Date: Wed, 21 Dec 2011 12:02:38 -0500
Subject: test plan, any suggestions?
Content:
 doc/testing.tex |   44 +++++++++++++++++++++++++++++++++++++++-----
 1 files changed, 39 insertions(+), 5 deletions(-)

b94609a: Robert Martin <rdm2128@columbia.edu>
Date: Wed, 21 Dec 2011 02:37:52 -0500
Subject: Update the introduction to more accurately reflect Clam as it currently exists
Content:
 doc/intro.tex |   23 ++++++++++++++---------
 1 files changed, 14 insertions(+), 9 deletions(-)

f8430a1: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 02:24:03 -0500
```

```
Subject: Moved Git history to separate appendix
Content:
 doc/appendix.tex |    4 ++++
 doc/paper.bib    |    6 ++++++
 2 files changed, 10 insertions(+), 0 deletions(-)

07c6473: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 01:45:37 -0500
Subject: nits/updates in tutorial
Content:
 doc/tutorial.tex |  141 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++----------------------------------------------------------
 1 files changed, 84 insertions(+), 57 deletions(-)

06f5fb9: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 01:03:06 -0500
Subject: Updated sobel source to reflect new Kernel syntax
Content:
 doc/src/sobel.clam |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

ea7f6fd: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 21 Dec 2011 00:59:37 -0500
Subject: Fixed single-calc assignment bug + Code Cleanup (for report listing)
Content: This fix slightly changed the language semantics (I updated all relevant
CLAM source tests/examples). You can now assign a single calc to a
Kernel, but all kernel assignments much start with the PIPE symbol.
Sample Kernel assignments:
    Kernel a = | someCalc1 | @someCalc2 | someCalc3;
    Kernel b = | someCalc2;
    Kernel c = | @someCalc3;

 clam/ast.mli                    |    8 ++++--
 clam/backend.ml                 |   43 +++++++++++++++++++++++++++++++++++++++++++----------------------------
 clam/clam.ml                    |    1 -
 clam/clamsys.ml                 |   13 +++++++++---
 clam/environ.ml                 |    6 +++-
 clam/envtypes.mli               |    3 +-
 clam/parser.mly                 |   17 ++++++++---------
 clam/parseutil.ml               |    5 +---
 clam/sast.mli                   |    2 +-
 clam/semantic.ml                |   20 +++++++++++++++++----
 clam/tests/1calc-ker.clam       |    2 +-
 clam/tests/addker.clam          |    4 +-
 clam/tests/addker2img.clam      |    4 +-
 clam/tests/at-channel.clam      |    2 +-
 clam/tests/clamsrc/blur.clam    |    5 ++-
 clam/tests/clamsrc/test.clam    |    4 +-
 clam/tests/convoperand.clam     |    2 +-
 clam/tests/imgwrite-norgb.clam  |    2 +-
 clam/tests/rval-conv.clam       |    2 +-
 clam/tests/rval-kernel.clam     |    2 +-
 clam/tests/sobel.clam           |    2 +-
 clam/verifier.ml                |    8 +++---
 22 files changed, 88 insertions(+), 69 deletions(-)

b1d1240: Kevin Sun <kfs2110@columbia.edu>
Date: Tue, 20 Dec 2011 23:53:20 -0500
Subject: Finished tutorial. Commented and reformatted sobel.clam (in docs/src/).
Content:
 doc/src/sobel.clam |   26 ++++++++-
 doc/tutorial.tex   |  160 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++------------------------------------
 2 files changed, 132 insertions(+), 54 deletions(-)

518fe66: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 23:25:00 -0500
Subject: Added all unit tests to source listing (+src listing cleanup)
Content: I no longer felt the need to list all the extlib and stb_image* sources.
I simply reference them and note that we used them. Now we only include
source listings for code we actually wrote.

 clam/tests/1calc-ker.clam          |    3 +
 clam/tests/1calc-ker.test          |   29 +++++
 clam/tests/1calc_ker.clam          |    3 -
 clam/tests/1calc_ker.test          |   29 -----
 clam/tests/at-channel.clam         |   17 +++
 clam/tests/at-channel.test         |   15 +++
 clam/tests/at_channel.clam         |   17 ---
 clam/tests/at_channel.test         |   15 ---
 clam/tests/equality-trans.clam     |    7 +
 clam/tests/equality-trans.test     |   30 +++++
 clam/tests/equality_trans.clam     |    7 -
```

```
clam/tests/equality_trans.test    |   30 -----
clam/tests/id-overlap.clam        |    3 +
clam/tests/id-overlap.test        |   27 +++++
clam/tests/id_overlap.clam        |    3 -
clam/tests/id_overlap.test        |   27 -----
clam/tests/image-defeq.clam       |    1 +
clam/tests/image-defeq.test       |   15 +++
clam/tests/image-eq-image.clam    |    2 +
clam/tests/image-eq-image.test    |   30 +++++
clam/tests/image-oreq-image.clam  |    3 +
clam/tests/image-oreq-image.test  |   29 +++++
clam/tests/image-oreq-image2.clam |    3 +
clam/tests/image-oreq-image2.test |   15 +++
clam/tests/image_defeq.clam       |    1 -
clam/tests/image_defeq.test       |   15 ---
clam/tests/image_eq_image.clam    |    2 -
clam/tests/image_eq_image.test    |   30 -----
clam/tests/image_oreq_image.clam  |    3 -
clam/tests/image_oreq_image.test  |   29 -----
clam/tests/image_oreq_image2.clam |    3 -
clam/tests/image_oreq_image2.test |   15 ---
clam/tests/imgread-bad.clam       |    2 +
clam/tests/imgread-bad.test       |   26 +++++
clam/tests/imgread-bad2.clam      |    1 +
clam/tests/imgread-bad2.test      |   15 +++
clam/tests/imgread-bad3.clam      |    1 +
clam/tests/imgread-bad3.test      |   15 +++
clam/tests/imgread_bad.clam       |    2 -
clam/tests/imgread_bad.test       |   26 -----
clam/tests/imgread_bad2.clam      |    1 -
clam/tests/imgread_bad2.test      |   15 ---
clam/tests/imgread_bad3.clam      |    1 -
clam/tests/imgread_bad3.test      |   15 ---
clam/tests/imgwrite-bad1.clam     |    1 +
clam/tests/imgwrite-bad1.test     |   15 +++
clam/tests/imgwrite-norgb.clam    |    6 +
clam/tests/imgwrite-norgb.test    |   15 +++
clam/tests/imgwrite_bad1.clam     |    1 -
clam/tests/imgwrite_bad1.test     |   15 ---
clam/tests/imgwrite_norgb.clam    |    6 -
clam/tests/imgwrite_norgb.test    |   15 ---
clam/tests/keyword-id.clam        |    1 +
clam/tests/keyword-id.test        |   15 +++
clam/tests/keyword_id.clam        |    1 -
clam/tests/keyword_id.test        |   15 ---
clam/tests/rval-calc.clam         |    2 +
clam/tests/rval-calc.test         |   29 +++++
clam/tests/rval-chanref.clam      |    3 +
clam/tests/rval-chanref.test      |   29 +++++
clam/tests/rval-conv.clam         |   13 ++
clam/tests/rval-conv.test         |   29 +++++
clam/tests/rval-cstr.clam         |    1 +
clam/tests/rval-cstr.test         |   29 +++++
clam/tests/rval-image.clam        |    2 +
clam/tests/rval-image.test        |   29 +++++
clam/tests/rval-imgread.clam      |    1 +
clam/tests/rval-imgread.test      |   29 +++++
clam/tests/rval-kernel.clam       |    6 +
clam/tests/rval-kernel.test       |   29 +++++
clam/tests/rval-matrix.clam       |    1 +
clam/tests/rval-matrix.test       |   29 +++++
clam/tests/rval_calc.clam         |    2 -
clam/tests/rval_calc.test         |   29 -----
clam/tests/rval_chanref.clam      |    3 -
clam/tests/rval_chanref.test      |   29 -----
clam/tests/rval_conv.clam         |   13 --
clam/tests/rval_conv.test         |   29 -----
clam/tests/rval_cstr.clam         |    1 -
clam/tests/rval_cstr.test         |   29 -----
clam/tests/rval_image.clam        |    2 -
clam/tests/rval_image.test        |   29 -----
clam/tests/rval_imgread.clam      |    1 -
clam/tests/rval_imgread.test      |   29 -----
clam/tests/rval_kernel.clam       |    6 -
clam/tests/rval_kernel.test       |   29 -----
clam/tests/rval_matrix.clam       |    1 -
clam/tests/rval_matrix.test       |   29 -----
doc/appendix.tex                  |  220 +++++++++++++++++++++++++++++++++++--
doc/paper.bib                     |   13 ++
90 files changed, 826 insertions(+), 613 deletions(-)
```

```
304afa9: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 22:21:08 -0500
Subject: Automatically generate a git-shortlog during "make"
Content:
 .gitignore              |    2 ++
 .mailmap                |    8 ++++++++
 doc/Makefile            |   12 ++++++++----
 doc/gen_gitshortlog.pl  |   24 ++++++++++++++++++++++++
 4 files changed, 42 insertions(+), 4 deletions(-)

13c7955: Kevin Sun <kfs2110@columbia.edu>
Date: Tue, 20 Dec 2011 21:49:43 -0500
Subject: Basic formatting for tutorial, still have to reorganize text. Put in some "lessons learned".
Content:
 doc/lessons-kevin.tex |   14 ++++++++++++++
 doc/src/sobel.clam    |   21 +++++++++++++++++++++
 doc/tutorial.tex      |   51 ++++++++++++++++++++++++------------------------
 3 files changed, 59 insertions(+), 27 deletions(-)

6f470f7: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 21:37:12 -0500
Subject: renamed some more underscore files... this gets old REALLY fast...
Content:
 clam/tests/_breakdown.sh            |    1 +
 clam/tests/_buildup.sh              |   12 +++---------
 clam/tests/breakdown-script-link.sh |    1 +
 clam/tests/build-up-script-link.sh  |    1 +
 4 files changed, 6 insertions(+), 9 deletions(-)

74788a8: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 21:13:37 -0500
Subject: renamed stb_image_* files to be nice to stupid Latex...
Content: WHY WHY WHY can't latex just handle underscore characters normally...

 clam/Makefile                  |    2 +-
 clam/clam.ml                   |    3 +-
 clam/libstb/stb-image-read.c   | 4734 ++++++++++++++++++++++++++++++++++++++++++
 clam/libstb/stb-image-write.c  |  583 +++++
 clam/libstb/stb_image_read.c   | 4734 ------------------------------------------
 clam/libstb/stb_image_write.c  |  583 -----
 doc/appendix.tex               |    4 +-
 7 files changed, 5322 insertions(+), 5321 deletions(-)

6be18c4: Kevin Sun <kfs2110@columbia.edu>
Date: Tue, 20 Dec 2011 21:11:23 -0500
Subject: Copied Tutorial text from presentation slides. To do: Fix formatting (headers/syntax coloring),
    and reorganize text to make more sense in context.
Content:
 doc/tutorial.tex |  116 ++++++++++++++++++++++++++++++++++++++++++++++++++++-------
 1 files changed, 100 insertions(+), 16 deletions(-)

fad9564: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 20:56:09 -0500
Subject: Structure for individual 'Lessons Learned' sections
Content:
 doc/lessons.tex         |   14 +++++++++++++-
 1 files changed, 13 insertions(+), 1 deletions(-)

8eebf5e: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 17:01:46 -0500
Subject: Initial Code Listing Appendix (still missing a few things)
Content:
 .gitignore           |    1 +
 doc/appendix.tex     |   62 +++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 doc/clamlisting.tex  |   13 ++++++----
 doc/ocamllisting.tex |   41 +++++++++++++++++++++++++++++++++++++++++
 doc/paper.bib        |    6 +++++
 doc/paper.tex        |    1 +
 6 files changed, 117 insertions(+), 7 deletions(-)

976c918: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 17:00:22 -0500
Subject: Updated formatting for code listing
Content:
 clam/clam.ml |   19 ++++++-------------
 1 files changed, 6 insertions(+), 13 deletions(-)

5d25ad5: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 16:15:19 -0500
Subject: Renamed parse_util.ml to parseutil.ml (to be nicer to Latex file include)
Content:
 clam/Makefile      |    2 +-
 clam/clam.ml       |    6 +++---
 clam/parse_util.ml |   49 -------------------------------------------------
 clam/parseutil.ml  |   49 +++++++++++++++++++++++++++++++++++++++++++++++++
 clam/printer.ml    |    2 +-
 5 files changed, 54 insertions(+), 54 deletions(-)

298dada: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 15:27:26 -0500
Subject: Starting with Edward's suggested sub-points in each chapter
Content:
 doc/design.tex  |    5 +++++
 doc/intro.tex   |    2 --
 doc/lessons.tex |    3 +++
 doc/plan.tex    |   14 +++++++++++++-
 doc/testing.tex |   10 ++++++++++
 5 files changed, 31 insertions(+), 3 deletions(-)

c39eab0: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 20 Dec 2011 09:31:56 -0500
Subject: Default to Final Report generation + fix Latex document build
Content:
 doc/Makefile        |   30 +++++++++++++++---------------
 doc/lrm-lexical.tex |    4 ++--
 doc/manual.tex      |    4 ++--
 doc/paper.bib       |    4 ++--
 doc/paper.tex       |    5 +++--
 doc/tutorial.tex    |    2 +-
 6 files changed, 25 insertions(+), 24 deletions(-)

c14161f: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 16:19:57 -0500
Subject: More formal integration of generated files
Content:
 clam/Makefile |   25 +++++++++++++------------
 1 files changed, 13 insertions(+), 12 deletions(-)

c2409f9: Kevin Sun <kfs2110@columbia.edu>
Date: Sun, 18 Dec 2011 16:09:26 -0500
Subject: Disallow " and ' in escaped C
Content:
 clam/scanner.mll |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

38fa689: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 15:44:13 -0500
Subject: Updated gitignore
Content:
 clam/.gitignore |    3 +++
 1 files changed, 3 insertions(+), 0 deletions(-)

cee04d7: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 14:25:08 -0500
Subject: more testing stuff
Content:
 clam/tests/clamsrc/blur.clam      |   27 +++++++++++++++++++++++++++
 clam/tests/clamsrc/subtract.clam  |   21 +++++++++++++++++++++
 clam/tests/clamsrc/test.clam      |   33 +++++++++++++++++++++++++++++++++
 clam/tests/clamsrc/writeread.clam |    5 +++++
 clam/tests/lena.jpg               |  Bin 0 -> 103932 bytes
 5 files changed, 86 insertions(+), 0 deletions(-)

c54831f: Kevin Sun <kfs2110@columbia.edu>
Date: Sun, 18 Dec 2011 12:11:51 -0500
Subject: Fixed image_eq_image.test to accept
Content:
 clam/tests/image_eq_image.test |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

c0a2386: Kevin Sun <kfs2110@columbia.edu>
Date: Sun, 18 Dec 2011 12:10:53 -0500
Subject: Fixed equality_trans.test to accept
Content:
 clam/tests/equality_trans.test |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

e5507af: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 10:08:15 -0500
Subject: Passes the 1 calc kernel test
Content:
 clam/semantic.ml |    6 +++++-
 clam/verifier.ml |   24 ++++++++++++++++--------
 2 files changed, 21 insertions(+), 9 deletions(-)

20e3a58: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 07:31:08 -0500
Subject: Fixed scanner bugs + added default R/G/B calc types
Content:
 clam/backend.ml       |    2 +-
 clam/environ.ml       |    5 +++--
 clam/scanner.mll      |   20 +++++++++-----------
 clam/semantic.ml      |   24 ------------------------
 clam/tests/sobel.clam |    4 ++--
 clam/verifier.ml      |    8 +++++---
 6 files changed, 20 insertions(+), 43 deletions(-)

286df92: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 05:56:08 -0500
Subject: Fixed multiply defined variables
Content:
 clam/environ.ml   |   14 +++++++++++++-
 clam/envtypes.mli |    1 +
 clam/semantic.ml  |    2 +-
 clam/verifier.ml  |    3 ++-
 4 files changed, 17 insertions(+), 3 deletions(-)

428a159: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 05:42:49 -0500
Subject: Added expected sobel output
Content:
 clam/tests/ucla_sobel_expected.png |  Bin 0 -> 116714 bytes
 1 files changed, 0 insertions(+), 0 deletions(-)

1c4f187: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 05:41:03 -0500
Subject: Append channels to the image properly
Content:
 clam/backend.ml  |    4 ----
 clam/verifier.ml |    5 +++--
 2 files changed, 3 insertions(+), 6 deletions(-)

7faa57f: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 05:37:17 -0500
Subject: Sobel calculation works!
Content:
 clam/backend.ml     |   50 ++++++++++++++++++++++++++++++-----------------------
 clam/environ.ml     |    5 ++++-
 clam/libstb/clam.c  |    1 -
 3 files changed, 31 insertions(+), 25 deletions(-)

00091ff: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 04:45:50 -0500
Subject: Sobel compiles
Content:
 clam/backend.ml  |   15 ++++++++++-----
 clam/sast.mli    |    2 +-
 clam/semantic.ml |    2 +-
 3 files changed, 12 insertions(+), 7 deletions(-)

f78393b: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 04:26:38 -0500
Subject: Partial kernel assignment
Content:
 clam/backend.ml     |   19 ++++++++++++++++---
 clam/libstb/clam.c  |    2 +-
 clam/libstb/clam.h  |    7 ++++---
 3 files changed, 21 insertions(+), 7 deletions(-)

5cce1e9: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 03:58:30 -0500
Subject: Added support for C-style casts in escaped-c strings
Content:
 clam/backend.ml  |    3 ++-
 clam/scanner.mll |    5 +++++
 2 files changed, 7 insertions(+), 1 deletions(-)

f78db29: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 03:57:29 -0500
Subject: Updated sobel example to be correct
Content:
 clam/tests/sobel.clam |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)
```

```
df5dbb3: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 03:32:18 -0500
Subject: Convolution function generation works
Content:
 clam/backend.ml    |   34 ++++++++++++++++++++++++++++++----
 clam/libstb/clam.h |    3 +++
 clam/sast.mli      |   17 +++++++++--------
 clam/semantic.ml   |    7 +++++--
 clam/verifier.ml   |    2 +-
 5 files changed, 48 insertions(+), 15 deletions(-)

90914c1: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 02:07:47 -0500
Subject: Convolution prep: functionalize
Content:
 clam/backend.ml                |   25 +++++++++++++++++--------
 clam/environ.ml                |    7 ++++---
 clam/envtypes.mli              |    9 ++++++++-
 clam/libstb/clam.c             |   28 ++++++++++++++++++++++++++++
 clam/libstb/clam.h             |    2 +-
 clam/libstb/stb_image_read.c   |    8 +++++---
 clam/libstb/stb_image_write.c  |    8 +++++---
 clam/libstb/template.c         |   35 ++++++++++++++++++++++++++++++++---
 clam/sast.mli                  |    2 +-
 clam/semantic.ml               |   11 +++++++++--
 clam/verifier.ml               |   18 ++++++++++--------
 11 files changed, 120 insertions(+), 33 deletions(-)

ae2551d: Kevin Sun <kfs2110@columbia.edu>
Date: Sun, 18 Dec 2011 01:46:03 -0500
Subject: Fixed imgchannel4.test
Content:
 clam/tests/imgchannel2.clam |    3 +--
 clam/tests/imgchannel4.clam |    4 +---
 clam/tests/imgchannel4.test |    3 ++-
 3 files changed, 4 insertions(+), 6 deletions(-)

e5f9735: Kevin Sun <kfs2110@columbia.edu>
Date: Sun, 18 Dec 2011 01:39:31 -0500
Subject: Changed imgchannel2.clam to match description
Content:
 clam/tests/imgchannel2.clam |    1 -
 1 files changed, 0 insertions(+), 1 deletions(-)

b1fb90d: Kevin Sun <kfs2110@columbia.edu>
Date: Sun, 18 Dec 2011 01:26:44 -0500
Subject: Reversed result of cstring3.test so accepting empty C-strings is good
Content:
 clam/tests/cstring3.test |   12 +++++++-----
 1 files changed, 7 insertions(+), 5 deletions(-)

da1419d: Kevin Sun <kfs2110@columbia.edu>
Date: Sun, 18 Dec 2011 01:23:18 -0500
Subject: Oops sorry was toying with _buildup.sh again, reverted now.
Content:
 clam/tests/_buildup.sh |    3 +--
 1 files changed, 1 insertions(+), 2 deletions(-)

93e216c: Kevin Sun <kfs2110@columbia.edu>
Date: Sun, 18 Dec 2011 01:09:10 -0500
Subject: Changed scanner to actually catch mismatched parens/calls in escaped C (Wasn't maintaining level
      correctly before) Two more tests pass now
Content:
 clam/scanner.mll               |   13 ++++++++-----
 clam/tests/1calc_ker.test      |    4 ++--
 clam/tests/_buildup.sh         |    3 ++-
 clam/tests/addker.clam         |    4 +---
 clam/tests/imgwrite_norgb.clam |    7 ++++---
 5 files changed, 17 insertions(+), 14 deletions(-)

aa3ced7: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 18 Dec 2011 00:13:31 -0500
Subject: Revert "Changed _buildup.sh (temporarily) so that failure for gcc to compile doesn't count as
     error"
Content: This reverts commit 92a6a5f14793510ed5da7c2ce7c604ceb59d38a4.

 clam/tests/_buildup.sh |    3 +--
 1 files changed, 1 insertions(+), 2 deletions(-)

312c123: Robert Martin <rdm2128@columbia.edu>
```

```
Date: Sun, 18 Dec 2011 00:09:19 -0500
Subject: Fix ordering of operands to g++ in clamsys
Content:
 clam/clamsys.ml |    6 +++---
 1 files changed, 3 insertions(+), 3 deletions(-)

6f051cb: Robert Martin <rdm2128@columbia.edu>
Date: Sun, 18 Dec 2011 00:08:52 -0500
Subject: Move definition of bail to clam.h
Content:
 clam/libstb/clam.c |    6 ------
 clam/libstb/clam.h |    6 ++++++
 2 files changed, 6 insertions(+), 6 deletions(-)

189ec8c: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 23:54:28 -0500
Subject: Changed keyword_id.clam to actually be meaningful. ("Red" technically isn't a keyword; "Kernel"
     certainly is.)
Content:
 clam/tests/keyword_id.clam |    2 +-
 clam/tests/keyword_id.test |    4 ++--
 2 files changed, 3 insertions(+), 3 deletions(-)

4b2a319: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 23:46:06 -0500
Subject: Temporarily changed tests/_buildup.sh to ignore GCC errors
Content:
 clam/tests/_buildup.sh |    3 ++-
 1 files changed, 2 insertions(+), 1 deletions(-)

0f751b0: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 23:35:01 -0500
Subject: Add backend for Calc Matrix -> Generated C code
Content:
 clam/backend.ml    |   31 +++++++++++++++++++++++++------
 clam/libstb/clam.h |    3 +--
 clam/sast.mli      |    4 ++--
 clam/semantic.ml   |   14 ++++++++------
 4 files changed, 36 insertions(+), 16 deletions(-)

92a6a5f: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 23:20:44 -0500
Subject: Changed _buildup.sh (temporarily) so that failure for gcc to compile doesn't count as error (
     Otherwise gets lots of false positives/negatives)
Content:
 clam/tests/_buildup.sh |    3 ++-
 1 files changed, 2 insertions(+), 1 deletions(-)

186ba35: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 23:02:04 -0500
Subject: Fix some minor bugs in clam.h
Content:
 clam/libstb/clam.h |    9 +++------
 1 files changed, 3 insertions(+), 6 deletions(-)

0648e4b: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 22:46:19 -0500
Subject: Add ability to change a sMatrix -> the { { } } C format
Content:
 clam/backend.ml  |   16 +++++++++++++++-
 clam/sast.mli    |    2 +-
 clam/semantic.ml |    4 +++-
 3 files changed, 19 insertions(+), 3 deletions(-)

98a5772: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 22:13:22 -0500
Subject: Add # of argument runtime check into a.out.
Content:
 clam/backend.ml  |    5 +++++
 clam/sast.mli    |    1 +
 clam/semantic.ml |    8 ++++++--
 3 files changed, 12 insertions(+), 2 deletions(-)

dcf8512: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 22:02:29 -0500
Subject: Improve Verifier handling of ImgRead and ImgWrite
Content: Verifier only accepts 1-parameter ImgRead -- a string or an int.
The int will represent an argv[x] value. ImgWrite accepts 3 parameters
which must be (image expression, string, string/int). The string/int
is again, either a filepath or an argv[x].
```

```
 clam/verifier.ml |   41 ++++++++++++++++++++++++++++++++++++++++-
 1 files changed, 40 insertions(+), 1 deletions(-)

10f9239: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 21:15:50 -0500
Subject: Add clam_clib.ml to PHONY so it always rebuilds
Content:
 clam/Makefile |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

1ef9eb5: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 21:01:56 -0500
Subject: Add a couple more stubs to clam.h and clam.c, slightly improve back-end code, fix a small bug in
     clam.h
Content:
 clam/backend.ml     |    9 ++++++---
 clam/libstb/clam.c |    4 ++++
 clam/libstb/clam.h |   11 ++++++-----
 3 files changed, 16 insertions(+), 8 deletions(-)

36684e2: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 20:26:32 -0500
Subject: Add kernel_copy to backend, fix bug in Makefile, add Null check to kernel_copy
Content:
 clam/Makefile       |    2 +-
 clam/backend.ml     |    8 +-------
 clam/libstb/clam.h |    7 ++++++-
 3 files changed, 8 insertions(+), 9 deletions(-)

f310e7a: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sat, 17 Dec 2011 18:18:01 -0500
Subject: Fixed backend bugs in channel assignment
Content:
 clam/backend.ml     |    4 ++--
 clam/libstb/clam.c |    6 +++++-
 clam/libstb/clam.h |   11 +++++++++--
 clam/verifier.ml    |    3 +--
 4 files changed, 17 insertions(+), 7 deletions(-)

24ac504: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sat, 17 Dec 2011 17:40:05 -0500
Subject: Auto C backend header generation
Content: ... and some other stuff

 clam/.gitignore         |    2 +
 clam/Makefile           |   16 ++-
 clam/backend.ml         |    2 +-
 clam/clamsys.ml         |    2 +-
 clam/libstb/clam.c      |  356 ++++++++++++++++++++++++++++++++++++++++++++++
 clam/libstb/clam.h      |   11 ++
 clam/libstb/template.c |  374 +----------------------------------------------
 clam/tests/ucla.png     |  Bin 112192 -> 144718 bytes
 8 files changed, 392 insertions(+), 371 deletions(-)

c7a7576: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sat, 17 Dec 2011 16:53:53 -0500
Subject: Convolution operator restrited to ChanRef * Id
Content:
 clam/ast.mli      |    4 ++--
 clam/parser.mly  |    2 +-
 clam/printer.ml  |    2 +-
 clam/sast.mli     |    2 +-
 clam/semantic.ml |    6 ++++++
 clam/verifier.ml |   11 ++++++++---
 6 files changed, 19 insertions(+), 8 deletions(-)

60e05b8: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 16:39:28 -0500
Subject: Add more C-string tests: open parentheses test and "sanity check" test that should pass. Both
    throw Backend errors right now. (which means the open parentheses isn't caught earlier)
Content:
 clam/tests/cstring5.clam |    1 +
 clam/tests/cstring5.test |   14 ++++++++++++++
 clam/tests/cstring6.clam |    1 +
 clam/tests/cstring6.test |   16 ++++++++++++++++
 4 files changed, 32 insertions(+), 0 deletions(-)

5d5d5fe: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 16:29:37 -0500
Subject: Add backend checking to distinguish between image 'assignments' and image 'copies'
Content:
 clam/backend.ml |   12 +++++++++++-
 1 files changed, 11 insertions(+), 1 deletions(-)

83ed9fa: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sat, 17 Dec 2011 16:26:05 -0500
Subject: Convolution works in the C-backend example.
Content:
 clam/libstb/build_template.sh |    4 +-
 clam/libstb/clam.h            |   55 +++++++++-------
 clam/libstb/template.c        |  143 ++++++++++++++++++++-------------------------
 3 files changed, 107 insertions(+), 95 deletions(-)

43eb75b: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 16:13:32 -0500
Subject: Start calc assignment, add "matrix" to environment
Content: I'm using a wrapper variable that holds the envT from the Verifier,
as well as accumulating read-only data. That way it won't affect
the verifier in any way.

 clam/backend.ml  |   16 ++++++----------
 clam/sast.mli    |    1 +
 clam/semantic.ml |    7 +++++--
 3 files changed, 12 insertions(+), 12 deletions(-)

1eba7bb: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 15:52:33 -0500
Subject: ImgRead added to backend
Content:
 clam/backend.ml |    4 +---
 1 files changed, 1 insertions(+), 3 deletions(-)

b16072f: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 15:49:06 -0500
Subject: Added checks for cstring validity. Getting some "C-Backend" errors at the moment.
Content:
 clam/tests/cstring1.clam |    1 +
 clam/tests/cstring1.test |   14 ++++++++++++++
 clam/tests/cstring2.clam |    1 +
 clam/tests/cstring2.test |   14 ++++++++++++++
 clam/tests/cstring3.clam |    1 +
 clam/tests/cstring3.test |   14 ++++++++++++++
 clam/tests/cstring4.clam |    1 +
 clam/tests/cstring4.test |   14 ++++++++++++++
 8 files changed, 60 insertions(+), 0 deletions(-)

cb9befc: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 15:47:22 -0500
Subject: Add more code-writing ability to backend
Content:
 clam/backend.ml |   43 +++++++++++++++++++++++++------------------
 1 files changed, 25 insertions(+), 18 deletions(-)

bcd75f6: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 15:34:26 -0500
Subject: (Backend seems to be in flux, will check back on this later.)
Content:
 clam/scanner.mll |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

f0b5fa1: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 15:23:43 -0500
Subject: Tests passed: disallow comments or preprocessor in C-String calc definition
Content:
 clam/parser.mly  |    7 -------
 clam/scanner.mll |    2 +-
 2 files changed, 1 insertions(+), 8 deletions(-)

1c6cb68: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 15:22:48 -0500
Subject: ImgWrite, ChanRef handled by backend
Content:
 clam/backend.ml |   20 ++++++++++++--------
 1 files changed, 12 insertions(+), 8 deletions(-)

a1b5ba5: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 15:12:41 -0500
Subject: Fixed matrix definition in parser to disallow bizarre definitions like: { 0 0 0 } 1 1 1 , 2 2 2
    } 3 3 3 ,
Content:
 clam/parser.mly |   17 +++++++++++++----
 1 files changed, 13 insertions(+), 4 deletions(-)
```

```
12781e0: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 15:07:33 -0500
Subject: Handle Image types internally as Enum instead of strings
Content:
 clam/backend.ml               |   47 +++++++++++++++++++++++++++++-----------
 clam/libstb/clam.h            |    9 +++++++-
 clam/libstb/stb_image_write.c |   19 +++++++++-------
 clam/sast.mli                 |    2 +-
 clam/semantic.ml              |    2 +
 5 files changed, 56 insertions(+), 23 deletions(-)

7627e8e: Kevin Sun <kfs2110@columbia.edu>
Date: Sat, 17 Dec 2011 14:57:00 -0500
Subject: Tests failed: Accepted ill-formed matrices. I'll fix this now.
Content:
 clam/tests/matrix3.clam |    1 +
 clam/tests/matrix3.test |   14 ++++++++++++++
 clam/tests/matrix4.clam |    1 +
 clam/tests/matrix4.test |   14 ++++++++++++++
 4 files changed, 30 insertions(+), 0 deletions(-)

1e0c418: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sat, 17 Dec 2011 14:38:09 -0500
Subject: Updated C backend structure
Content:
 clam/libstb/clam.h     |  173 ++++++++++++++++-----
 clam/libstb/template.c |  403 ++++++++++++++++++++++++++++++++++++++----------
 2 files changed, 458 insertions(+), 118 deletions(-)

68fcd37: Robert Martin <rdm2128@columbia.edu>
Date: Sat, 17 Dec 2011 00:26:33 -0500
Subject: Further streamline code in Semantic, add more hooks in backend.
Content: I think if GCC could read my comments and translate them into
C code, then Clam would be working right now. Since it doesn't,
I guess we'll have to replace the /* comments */ with actual C
code.

 clam/backend.ml  |   57 +++++++++++++++++++++++++++++++++++++++++++-----------
 clam/sast.mli    |    6 ++--
 clam/semantic.ml |   26 ++++++++++++------------
 3 files changed, 61 insertions(+), 28 deletions(-)

be26cf3: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Fri, 16 Dec 2011 15:54:43 -0500
Subject: A little more detail on C-style convolution backend
Content:
 clam/libstb/clam.h     |   25 +++++++++---------
 clam/libstb/template.c |   65 +++++++++++++++++++++++++++++++++++++++++++-----
 2 files changed, 70 insertions(+), 20 deletions(-)

613f1da: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Fri, 16 Dec 2011 15:12:39 -0500
Subject: Updated C-backend template: still missing convolution imp...
Content:
 clam/libstb/clam.h           |  129 ++++++++++++++++++++++-------------
 clam/libstb/stb_image_read.c |    6 +-
 clam/libstb/template.c       |  153 ++++++++++++++++++++++--------------------
 3 files changed, 163 insertions(+), 125 deletions(-)

a6c2f32: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 14:51:42 -0500
Subject: Finish (I think) the conversion from Ast -> Sast. Backend has some stubs.
Content: We can start fleshing out the backend more, or begin putting some C
flesh on the backend function skeleton.

We'll want to think about how variables get passed around in our C,
but I think the best way is to do what Edwards talked about in class
and create a stack.

 clam/semantic.ml |   32 +++++++++++++++-----------------
 1 files changed, 15 insertions(+), 17 deletions(-)

e2b8b64: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 14:26:13 -0500
Subject: Further improve backend and semantic Sast
Content:
 clam/backend.ml  |   34 ++++++++++++++++++++++++++--------
 clam/semantic.ml |   42 ++++++++++++++++--------------------------
 2 files changed, 42 insertions(+), 34 deletions(-)
```

```
76b7d17: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 14:05:48 -0500
Subject: Begin traversing the Sast tree in backend
Content:
 clam/backend.ml |   26 ++++++++++++++++++++++++--
 1 files changed, 24 insertions(+), 2 deletions(-)

4e6dcdd: Yongxu Zhang <yz2419@columbia.edu>
Date: Fri, 16 Dec 2011 14:01:04 -0500
Subject: some unit tests discussed in the morning
Content:
 clam/tests/1calc_ker.clam   |    3 +++
 clam/tests/1calc_ker.test   |   29 +++++++++++++++++++++++++++++
 clam/tests/DefEq.clam       |    1 -
 clam/tests/addker.clam      |   11 +++++++++++
 clam/tests/addker.test      |   29 +++++++++++++++++++++++++++++
 clam/tests/addker2img.clam  |   11 +++++++++++
 clam/tests/addker2img.test  |   15 +++++++++++++++
 clam/tests/cimage.clam      |    2 ++
 clam/tests/cimage.test      |   15 +++++++++++++++
 clam/tests/ckernel.clam     |    2 ++
 clam/tests/ckernel.test     |   15 +++++++++++++++
 clam/tests/convoperand.clam |    8 ++++++++
 clam/tests/convoperand.test |   15 +++++++++++++++
 clam/tests/keyword_id.clam  |    1 +
 clam/tests/keyword_id.test  |   15 +++++++++++++++
 15 files changed, 171 insertions(+), 1 deletions(-)

7666a0c: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:59:33 -0500
Subject: Changed imgwrite_norgb to write a file with only R and B
Content:
 clam/tests/imgwrite_norgb.clam |   12 +++++-------
 1 files changed, 5 insertions(+), 7 deletions(-)

29b1c70: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 13:58:19 -0500
Subject: Minor cleaning-up in backend.ml
Content:
 clam/backend.ml |    5 +----
 1 files changed, 1 insertions(+), 4 deletions(-)

290901b: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:57:53 -0500
Subject: Test failed: imgwrite ran with bad arguments
Content:
 clam/tests/imgwrite_bad1.clam |    1 +
 clam/tests/imgwrite_bad1.test |   15 +++++++++++++++
 2 files changed, 16 insertions(+), 0 deletions(-)

a0d4eea: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 13:54:31 -0500
Subject: Disallow VExpr from being a filename or an image format
Content:
 clam/backend.ml  |    1 -
 clam/sast.mli    |    2 --
 clam/semantic.ml |    2 --
 3 files changed, 0 insertions(+), 5 deletions(-)

8d3f60b: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 13:53:20 -0500
Subject: Add some basic structure to the backend to handle the Sast
Content:
 clam/backend.ml |  185 +++++++++++-------------------------------------------------
 1 files changed, 38 insertions(+), 147 deletions(-)

1f063b1: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:52:46 -0500
Subject: Tests failed: imgread accepts multiple arguments, and invalid argument types
Content:
 clam/tests/imgread_bad2.clam |    1 +
 clam/tests/imgread_bad2.test |   15 +++++++++++++++
 clam/tests/imgread_bad3.clam |    1 +
 clam/tests/imgread_bad3.test |   15 +++++++++++++++
 4 files changed, 32 insertions(+), 0 deletions(-)

82bdcc7: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:47:52 -0500
Subject: Test failed: wrote image without valid RGB channels
Content:
 clam/tests/imgwrite_norgb.clam |    7 +++++++
 clam/tests/imgwrite_norgb.test |   15 +++++++++++++++
 2 files changed, 22 insertions(+), 0 deletions(-)

64a02b3: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 13:38:23 -0500
Subject: Translate matrices into 'int list list's rather than 'BInt list list's
Content:
 clam/semantic.ml |   11 +++--------
 1 files changed, 3 insertions(+), 8 deletions(-)

e3d5311: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:36:30 -0500
Subject: Test succeeded: Equality is transferred (Image a = b = imgread(...))
Content:
 clam/tests/equality_trans.clam |    7 +++++++
 clam/tests/equality_trans.test |   29 +++++++++++++++++++++++++++++
 2 files changed, 36 insertions(+), 0 deletions(-)

0bdf902: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 13:36:26 -0500
Subject: Change all "Failure" exceptions in Semantic to "SemanticFailure" exceptions
Content:
 clam/clam.ml     |    1 +
 clam/semantic.ml |   58 +++++++++++++++++++++++++++++++----------------------------
 2 files changed, 31 insertions(+), 28 deletions(-)

bf9702a: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:33:04 -0500
Subject: Changed/added tests for image |= image (doesn't work, or distinguish size)
Content:
 clam/tests/id_overlap.clam         |    1 -
 clam/tests/image_oreq_image.clam   |    3 ++-
 clam/tests/image_oreq_image.test   |   26 ++++++++++++++++++++------
 clam/tests/image_oreq_image2.clam  |    3 +++
 clam/tests/image_oreq_image2.test  |   15 +++++++++++++++
 5 files changed, 40 insertions(+), 8 deletions(-)

70262b2: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 13:31:43 -0500
Subject: Prune some extra code from Semantic
Content:
 clam/semantic.ml |   12 ++++++------
 1 files changed, 6 insertions(+), 6 deletions(-)

244d3a7: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 13:26:26 -0500
Subject: Modify Sast stuff to use the Environment built in the verifier.
Content: Also, to prevent circular dependency, I moved the type declarations from
environ.ml into envtypes.mli.

 clam/Makefile     |    4 +-
 clam/backend.ml   |   12 ++---
 clam/clam.ml      |    6 +-
 clam/env.ml       |  156 ----------------------------------------------------
 clam/environ.ml   |   28 +---------
 clam/envtypes.mli |   42 ++++++++++++++
 clam/printer.ml   |   20 +++----
 clam/sast.mli     |    6 +-
 clam/semantic.ml  |   57 ++++++++++----------
 clam/verifier.ml  |    1 +
 10 files changed, 92 insertions(+), 240 deletions(-)

cffbf1c: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:24:55 -0500
Subject: Added two other images for testing
Content:
 clam/tests/flatiron.jpg |  Bin 0 -> 130593 bytes
 clam/tests/ucla2.png    |  Bin 0 -> 112192 bytes
 2 files changed, 0 insertions(+), 0 deletions(-)

8d895f5: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:22:32 -0500
Subject: Test failed: assign channels of different sizes
Content:
 clam/tests/sizediff.clam |    3 +++
 clam/tests/sizediff.test |   15 +++++++++++++++
 2 files changed, 18 insertions(+), 0 deletions(-)

c3be507: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:18:15 -0500
Subject: Test failed: undefined images still have default channels
```

```
Content:
 clam/tests/defchannels.clam |    2 ++
 clam/tests/defchannels.test |   15 +++++++++++++++
 2 files changed, 17 insertions(+), 0 deletions(-)

075b546: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 13:13:55 -0500
Subject: Added "r-value" tests: variable and operations work even if return values are ignored
Content:
 clam/tests/rval_calc.clam    |    2 ++
 clam/tests/rval_calc.test    |   29 +++++++++++++++++++++++++++++
 clam/tests/rval_chanref.clam |    3 +++
 clam/tests/rval_chanref.test |   29 +++++++++++++++++++++++++++++
 clam/tests/rval_conv.clam    |   13 +++++++++++++
 clam/tests/rval_conv.test    |   29 +++++++++++++++++++++++++++++
 clam/tests/rval_cstr.clam    |    1 +
 clam/tests/rval_cstr.test    |   29 +++++++++++++++++++++++++++++
 clam/tests/rval_image.clam   |    2 ++
 clam/tests/rval_image.test   |   29 +++++++++++++++++++++++++++++
 clam/tests/rval_imgread.clam |    1 +
 clam/tests/rval_imgread.test |   29 +++++++++++++++++++++++++++++
 clam/tests/rval_kernel.clam  |    6 ++++++
 clam/tests/rval_kernel.test  |   29 +++++++++++++++++++++++++++++
 clam/tests/rval_matrix.clam  |    1 +
 clam/tests/rval_matrix.test  |   29 +++++++++++++++++++++++++++++
 16 files changed, 261 insertions(+), 0 deletions(-)

39ce5ee: Kevin Sun <kfs2110@columbia.edu>
Date: Fri, 16 Dec 2011 12:54:14 -0500
Subject: Test for @ operator
Content:
 clam/tests/at_channel.clam |   17 +++++++++++++++++
 clam/tests/at_channel.test |   15 +++++++++++++++
 2 files changed, 32 insertions(+), 0 deletions(-)

05010b9: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Fri, 16 Dec 2011 12:03:36 -0500
Subject: Compress translation
Content:
 clam/semantic.ml |   11 +----------
 1 files changed, 1 insertions(+), 10 deletions(-)

f24d24d: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 11:41:35 -0500
Subject: Fix bug where only CString was accepted as an R-Value for |= operator
Content: ... forgot to commit this earlier, sorry.

 clam/semantic.ml |    7 ++-----
 1 files changed, 2 insertions(+), 5 deletions(-)

61e6974: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Fri, 16 Dec 2011 11:47:06 -0500
Subject: remove debugging
Content:
 clam/verifier.ml |    7 ++-----
 1 files changed, 2 insertions(+), 5 deletions(-)

2cdb3a4: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Fri, 16 Dec 2011 11:45:13 -0500
Subject: reverse the output of the verifier
Content:
 clam/verifier.ml |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

a37d9a7: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Fri, 16 Dec 2011 11:30:28 -0500
Subject: Small bugs in post-merge code
Content:
 clam/environ.ml     |    1 -
 clam/tests/all.test |    7 +++++++
 clam/verifier.ml    |   16 ++++++++--------
 3 files changed, 15 insertions(+), 9 deletions(-)

bbe74eb: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 03:11:03 -0500
Subject: Only allow CStrings to be OrEq with images or kernels
Content:
 clam/semantic.ml |   12 ++++++++----
 1 files changed, 8 insertions(+), 4 deletions(-)

53f3f6e: Robert Martin <rdm2128@columbia.edu>
```

```
Date: Fri, 16 Dec 2011 03:07:00 -0500
Subject: Improve matrix semantic checking
Content:
 clam/env.ml      |   11 +++++++++++
 clam/sast.mli    |    6 +++++-
 clam/semantic.ml |   33 +++++++++++++++++++++++++++------
 3 files changed, 43 insertions(+), 7 deletions(-)

5e3cb2a: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 02:12:01 -0500
Subject: Change tests because this branch doesn't allow "Calc" to be declared without begin defined
Content:
 clam/tests/defcalc2.clam    |    2 +-
 clam/tests/defcalc2.test    |   15 +++------------
 clam/tests/defcalc3.clam    |    3 +++
 clam/tests/defcalc3.test    |   20 ++++++++++++++++++++
 clam/tests/imgchannel2.clam |    2 +-
 5 files changed, 28 insertions(+), 14 deletions(-)

fd025f4: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 01:53:00 -0500
Subject: Unify things so they're pretty. We pass 13 tests.
Content:
 clam/Makefile |    2 +-
 clam/clam.ml  |   14 ++++++++------
 2 files changed, 9 insertions(+), 7 deletions(-)

db11f5d: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 01:46:30 -0500
Subject: Remove environ.ml and verifier.ml to try and help merge
Content:
 clam/verifier.ml |    4 ----
 1 files changed, 0 insertions(+), 4 deletions(-)

03f5492: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 01:44:08 -0500
Subject: Further prepare for the merge. Things seems to work at this point
Content:
 clam/Makefile    |    2 +-
 clam/backend.ml  |    2 +-
 clam/env.ml      |  145 ++++++++++++++++++++++++++++++++++++++++++++++++++++++
 clam/environ.ml  |  145 ------------------------------------------------------
 clam/semantic.ml |    2 +-
 clam/verifier.ml |    4 ++
 6 files changed, 152 insertions(+), 148 deletions(-)

4e9ee62: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 01:39:14 -0500
Subject: Move verifier.ml to semantic.ml. Getting ready to pull verifier
Content:
 clam/Makefile    |    4 +-
 clam/backend.ml  |    2 +-
 clam/clam.ml     |    5 +-
 clam/environ.ml  |    2 +-
 clam/printer.ml  |    2 +-
 clam/sast.mli    |   90 +++++++++++++++++++++
 clam/semantic.ml |  234 ++++++++++++++++++++++++++++++++++++++++++++++++++++++
 clam/vast.mli    |   90 ---------------------
 clam/verifier.ml |  234 ------------------------------------------------------
 9 files changed, 332 insertions(+), 331 deletions(-)

03640bb: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 01:32:39 -0500
Subject: Allow environment to affected by ChanRefs
Content:
 clam/environ.ml |   86 ++++++++++++++++++++++++++++++++++++++++++------------
 clam/printer.ml |    2 +
 2 files changed, 69 insertions(+), 19 deletions(-)

5d92d9e: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 00:48:53 -0500
Subject: Verified Environment can detect lvalues and rvalues
Content:
 clam/environ.ml  |   72 ++++++++++++++++++++++++++++-------------------------
 clam/verifier.ml |   23 +++++++++-------
 2 files changed, 51 insertions(+), 44 deletions(-)

668b9e3: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 16 Dec 2011 00:31:16 -0500
Subject: Big improvement to producing a much nicer AST
Content:
 clam/vast.mli    |    2 +
 clam/verifier.ml |   61 +++++++++++++++++++++++++++++++++++++++++++++++++++++----
 2 files changed, 58 insertions(+), 5 deletions(-)

7d8b891: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 15 Dec 2011 23:59:04 -0500
Subject: Disallow channel assignment from uninitialized CalcT
Content:
 clam/verifier.ml |   22 +++++++++++++++++-----
 1 files changed, 17 insertions(+), 5 deletions(-)

891ab7f: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 15 Dec 2011 23:51:29 -0500
Subject: Print error message from compile_it
Content:
 clam/tests/_buildup.sh |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

6fb3fb9: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 23:44:12 -0500
Subject: Added ability to parse Channel References
Content:
 clam/environ.ml  |    9 +++++++++
 clam/verifier.ml |   27 ++++++++++++++++++++++-----
 2 files changed, 31 insertions(+), 5 deletions(-)

7b64a41: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 15 Dec 2011 23:42:41 -0500
Subject: Checking for ImageT |= CalcT(matrix)
Content: The verifier now catches the catches image channel assignments from
matrix calculations (and throws an error)

 clam/ast.mli     |    2 -
 clam/environ.ml  |   18 ++++++------
 clam/verifier.ml |   74 ++++++++++++++++++++++++++++++++++++++++++------------------
 3 files changed, 66 insertions(+), 28 deletions(-)

0901f44: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 23:20:41 -0500
Subject: Verifier has a very limited set of things it can verify
Content:
 clam/environ.ml  |    9 +++++++
 clam/vast.mli    |   23 +++++-------------
 clam/verifier.ml |   67 +++++++++++++++++++++++++++++++++++++++++++++++++++++--
 3 files changed, 80 insertions(+), 19 deletions(-)

582c72f: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 15 Dec 2011 22:52:56 -0500
Subject: Add default channel names to an image
Content:
 clam/environ.ml |   36 ++++++++++++++++++++++++++++++++----
 clam/printer.ml |    5 -----
 2 files changed, 32 insertions(+), 9 deletions(-)

4e0da49: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 15 Dec 2011 22:17:45 -0500
Subject: More OCaml-like equal-row verification
Content:
 clam/verifier.ml |   22 ++++++++++++----------
 1 files changed, 12 insertions(+), 10 deletions(-)

b9a4d4d: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 22:14:53 -0500
Subject: Add "Unknown" atomic type for some Calc's. Switched so every expression parses as a Calc. Still
    doesn't verify anything
Content:
 clam/ast.mli     |    2 +-
 clam/printer.ml  |    1 +
 clam/vast.mli    |    8 ++++++++
 clam/verifier.ml |   14 ++++++++++----
 4 files changed, 20 insertions(+), 5 deletions(-)

0488de7: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 21:48:01 -0500
Subject: Verifier starting to work, but it currently thinks every expression is an ImageT.
Content: Obviously this causes some type conflicts....

 clam/backend.ml  |    2 +-
 clam/clam.ml     |    2 +-
 clam/environ.ml  |    9 +++++----
 clam/printer.ml  |   16 ++++++++++++++++
 clam/verifier.ml |   47 ++++++++++++++++++++++++++++++-----------------
 5 files changed, 53 insertions(+), 23 deletions(-)

2722e11: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 21:34:00 -0500
Subject: Outline for tutorial. Copied intro from proposal into Intro for final report.
Content:
 doc/intro.tex    |   31 +++++++++++++++++++++++++++++++
 doc/tutorial.tex |   32 ++++++++++++++++++++++++++++++++
 2 files changed, 63 insertions(+), 0 deletions(-)

d0464a1: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 21:10:58 -0500
Subject: Clam compiles. Sobel parses but is not being very strongly verified. Everything that gets
     generated can be understood by backend
Content:
 clam/ast.mli      |    1 -
 clam/backend.ml   |    2 +-
 clam/environ.ml   |   36 +++---
 clam/printer.ml   |    1 -
 clam/verifier.ml  |  325 +++++++-----------------------------------------------
 5 files changed, 60 insertions(+), 305 deletions(-)

f9cf117: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 15 Dec 2011 21:05:49 -0500
Subject: Replaced tab characters with spaces... sorry, I'm anal like that
Content:
 clam/verifier.ml |   18 +++++++++---------
 1 files changed, 9 insertions(+), 9 deletions(-)

b05b7a2: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 15 Dec 2011 21:04:57 -0500
Subject: ImageT() -> CalcT in '=' error message
Content:
 clam/verifier.ml |    6 +++---
 1 files changed, 3 insertions(+), 3 deletions(-)

35a0c4b: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 20:53:20 -0500
Subject: Removed reference to $() in LRM, since it's no longer needed
Content:
 doc/lrm-expressions.tex |   10 ----------
 1 files changed, 0 insertions(+), 10 deletions(-)

4a84733: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 20:46:05 -0500
Subject: Modified verifier to check that matrix rows are of equal length
Content:
 clam/verifier.ml |    9 ++++++++-
 1 files changed, 8 insertions(+), 1 deletions(-)

5579449: Yongxu Zhang <yz2419@columbia.edu>
Date: Thu, 15 Dec 2011 20:24:11 -0500
Subject: Calc should not be assigned by another Calc, so remove typeconvertion.test
Content:
 clam/tests/typeconvertion.clam |    3 ---
 clam/tests/typeconvertion.test |   15 ---------------
 2 files changed, 0 insertions(+), 18 deletions(-)

fea251b: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 20:10:28 -0500
Subject: Changed tests to use assignment operators properly
Content:
 clam/tests/imgchannel3.clam    |    2 +-
 clam/tests/imgchannel4.clam    |    2 +-
 clam/tests/matrix1.clam        |    2 +-
 clam/tests/matrix2.clam        |    2 +-
 clam/tests/typeconvertion.clam |    6 +++---
 5 files changed, 7 insertions(+), 7 deletions(-)

7423e3f: Yongxu Zhang <yz2419@columbia.edu>
Date: Thu, 15 Dec 2011 20:08:42 -0500
Subject: Added type checking for assignment operators with Calc
Content:
 clam/verifier.ml |   11 ++++++++++-
 1 files changed, 10 insertions(+), 1 deletions(-)

618a241: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 20:06:39 -0500
Subject: Verifier can recognize some declarations and backend can start to recognize verified AST
Content:
 clam/ast.mli      |    1 -
 clam/environ.ml   |   78 ++++++++++++++++++++++++++++++----------------------------
 clam/printer.ml   |    3 +-
 clam/vast.mli     |    9 ++----
 clam/verifier.ml  |   17 +++++++++---
 5 files changed, 56 insertions(+), 52 deletions(-)

54f444b: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 19:45:17 -0500
Subject: Added tests for image with various assignment operators (:=, =, |=)
Content:
 clam/tests/image_defeq.clam       |    1 +
 clam/tests/image_defeq.test       |   15 +++++++++++++++
 clam/tests/image_eq_image.clam    |    2 ++
 clam/tests/image_eq_image.test    |   29 +++++++++++++++++++++++++++++
 clam/tests/image_oreq_image.clam  |    2 ++
 clam/tests/image_oreq_image.test  |   15 +++++++++++++++
 6 files changed, 64 insertions(+), 0 deletions(-)

051107d: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 19:20:34 -0500
Subject: Create skeleton for verifier and backend to create a nice VAST instead of the AST
Content:
 clam/backend.ml   |   24 +++++++++++++++++++++++-
 clam/clam.ml      |    4 ++--
 clam/vast.mli     |   19 +++++++++++++++++--
 clam/verifier.ml  |   26 ++++++++++++++++++++++++++
 4 files changed, 68 insertions(+), 5 deletions(-)

3d17da6: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 17:50:22 -0500
Subject: Edited parser to accept matrices without scalers in front. ( [1 / 1] is the default )
Content:
 clam/parser.mly              |    1 +
 clam/tests/imagesize.clam    |    5 -----
 clam/tests/imagesize.test    |   15 ---------------
 clam/tests/imgchannel3.clam  |    2 +-
 4 files changed, 2 insertions(+), 21 deletions(-)

5c4d547: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 17:43:49 -0500
Subject: Added matrix-related tests: matrix1.test: Matrices without scaling factors are not accepted.
     matrix2.test: Matrices with unequal row length are accepted.???
Content:
 clam/tests/matrix1.clam         |    1 +
 clam/tests/matrix1.test         |   16 ++++++++++++++++
 clam/tests/matrix2.clam         |    1 +
 clam/tests/matrix2.test         |   14 ++++++++++++++
 clam/tests/typeconvertion.test  |    2 +-
 5 files changed, 33 insertions(+), 1 deletions(-)

fa73d4f: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 17:32:10 -0500
Subject: Change structure of environment variables, tweak vast, change how printer works
Content:
 clam/environ.ml  |  170 +++++++++++++++++++------------------------------------------
 clam/printer.ml  |   22 ++-----
 clam/vast.mli    |   32 +++++++++--
 3 files changed, 91 insertions(+), 133 deletions(-)

32a263c: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 17:24:15 -0500
Subject: Added more tests for adding Calcs to Images
Content:
 clam/tests/imgchannel3.clam  |    3 +++
 clam/tests/imgchannel3.test  |   26 ++++++++++++++++++++++++++
 clam/tests/imgchannel4.clam  |    5 +++++
 clam/tests/imgchannel4.test  |   28 ++++++++++++++++++++++++++++
 4 files changed, 62 insertions(+), 0 deletions(-)

70eb56e: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 17:23:01 -0500
Subject: Modified imgchannel2.test
Content:
 clam/tests/cstrdef.clam      |    5 -----
 clam/tests/cstrdef.test      |   28 ----------------------------
 clam/tests/imgchannel2.test  |   11 ++++-------
 3 files changed, 4 insertions(+), 40 deletions(-)

96e985a: Yongxu Zhang <yz2419@columbia.edu>
Date: Thu, 15 Dec 2011 17:21:51 -0500
```

```
Subject: An Eq, besides DefEq, can define a Calc
Content:
 clam/tests/DefEq.clam |    2 ++
 clam/tests/DefEq.test |   15 +++++++++++++++
 2 files changed, 17 insertions(+), 0 deletions(-)

a6eec2c: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 17:06:55 -0500
Subject: Clarified description of "imgchannel1.test"
Content:
 clam/tests/imgchannel1.test |    2 +-
 clam/tests/zerocalc2.test   |    9 +++------
 2 files changed, 4 insertions(+), 7 deletions(-)

784ce72: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 17:03:51 -0500
Subject: Corrected "cstrdef.test"
Content:
 clam/tests/cstrdef.test |    8 +++++---
 1 files changed, 5 insertions(+), 3 deletions(-)

bc5171d: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 16:57:27 -0500
Subject: Added tests for #[]# operators, and more channel manipulation stuff
Content:
 clam/tests/cstrdef.clam     |    5 +++++
 clam/tests/cstrdef.test     |   26 ++++++++++++++++++++++++++
 clam/tests/imgchannel2.clam |    5 +++++
 clam/tests/imgchannel2.test |   29 +++++++++++++++++++++++++++++
 4 files changed, 65 insertions(+), 0 deletions(-)

bb0aa49: Yongxu Zhang <yz2419@columbia.edu>
Date: Thu, 15 Dec 2011 16:52:28 -0500
Subject: correct the  descriptions of test files
Content:
 clam/tests/imagesize.test      |    2 +-
 clam/tests/typeconvertion.test |    2 +-
 2 files changed, 2 insertions(+), 2 deletions(-)

c2c509a: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 16:24:29 -0500
Subject: Changed verifier to catch divide-by-zero error in matrix definition. Is this right?
Content:
 clam/verifier.ml |    3 ++-
 1 files changed, 2 insertions(+), 1 deletions(-)

f80ee71: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 16:00:20 -0500
Subject: Added tests for overlapping identifiers, and channel manipulation
Content:
 clam/tests/id_overlap.clam  |    4 ++++
 clam/tests/id_overlap.test  |   27 +++++++++++++++++++++++++++
 clam/tests/imgchannel1.clam |    3 +++
 clam/tests/imgchannel1.test |   29 +++++++++++++++++++++++++++++
 4 files changed, 63 insertions(+), 0 deletions(-)

bb44b7b: Yongxu Zhang <yz2419@columbia.edu>
Date: Thu, 15 Dec 2011 15:44:10 -0500
Subject: an image could be assigned with a smaller matrix
Content:
 clam/tests/imagesize.clam |    5 +++++
 clam/tests/imagesize.test |   15 +++++++++++++++
 2 files changed, 20 insertions(+), 0 deletions(-)

3fa486b: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 15:16:15 -0500
Subject: Add structure for verified syntax tree (vast.mli)
Content:
 clam/Makefile |    2 +-
 clam/vast.mli |   55 +++++++++++++++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 56 insertions(+), 1 deletions(-)

fe057d4: Yongxu Zhang <yz2419@columbia.edu>
Date: Thu, 15 Dec 2011 15:10:23 -0500
Subject: variables could be assigned with shorter uint
Content:
 clam/tests/typeconvertion.clam |    3 +++
 clam/tests/typeconvertion.test |   15 +++++++++++++++
 2 files changed, 18 insertions(+), 0 deletions(-)

fb1cd44: Kevin Sun <kfs2110@columbia.edu>
```

```
Date: Thu, 15 Dec 2011 14:58:48 -0500
Subject: Added tests for multiplier [0 / 1] and [1 / 0]
Content:
 clam/tests/zerocalc1.clam |    1 +
 clam/tests/zerocalc1.test |   29 +++++++++++++++++++++++++++++
 clam/tests/zerocalc2.clam |    1 +
 clam/tests/zerocalc2.test |   29 +++++++++++++++++++++++++++++
 4 files changed, 60 insertions(+), 0 deletions(-)

a4f1dac: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 14:50:02 -0500
Subject: Renamed defcalc.* to defcalc1.*
Content:
 clam/tests/defcalc.clam  |    1 -
 clam/tests/defcalc.test  |   29 -----------------------------
 clam/tests/defcalc1.clam |    1 +
 clam/tests/defcalc1.test |   29 +++++++++++++++++++++++++++++
 4 files changed, 30 insertions(+), 30 deletions(-)

a3d3462: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 14:45:27 -0500
Subject: Added some tests for definition of Calc
Content:
 clam/tests/defcalc.clam  |    1 +
 clam/tests/defcalc.test  |   29 +++++++++++++++++++++++++++++
 clam/tests/defcalc2.clam |    2 ++
 clam/tests/defcalc2.test |   29 +++++++++++++++++++++++++++++
 4 files changed, 61 insertions(+), 0 deletions(-)

1488cf3: Kevin Sun <kfs2110@columbia.edu>
Date: Thu, 15 Dec 2011 14:33:09 -0500
Subject: Added some tests for imgread - must declare before assignment
Content:
 clam/tests/imgread.clam     |    2 ++
 clam/tests/imgread.test     |   29 +++++++++++++++++++++++++++++
 clam/tests/imgread_bad.clam |    2 ++
 clam/tests/imgread_bad.test |   26 ++++++++++++++++++++++++++
 4 files changed, 59 insertions(+), 0 deletions(-)

e56478d: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 13:29:06 -0500
Subject: Replace JPG with PNG, because our image library does not support JPG
Content:
 clam/tests/README          |    2 +-
 clam/tests/imgcopy.clam    |    4 ++--
 clam/tests/imgcopy.test    |    4 ++--
 clam/tests/sobel.clam      |    4 ++--
 clam/tests/sobel.test      |    6 +++---
 clam/tests/ucla.jpg        |  Bin 56626 -> 0 bytes
 clam/tests/ucla.png        |  Bin 0 -> 112192 bytes
 clam/tests/undefined1.clam |    2 +-
 8 files changed, 11 insertions(+), 11 deletions(-)

673b9a0: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 13:21:47 -0500
Subject: Additional structure for backend.ml, code now makes printfs instead of actually doing anything
    ....
Content:
 clam/backend.ml |   91 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++----------
 1 files changed, 75 insertions(+), 16 deletions(-)

4474e8c: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 12:19:15 -0500
Subject: Fix bug where declarations were in the wrong order
Content:
 clam/backend.ml |   15 ++++++++++-----
 1 files changed, 10 insertions(+), 5 deletions(-)

0d3175c: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 15 Dec 2011 12:12:34 -0500
Subject: Improve backend structure, add some stubs where code will need to be generated
Content:
 clam/backend.ml |   85 ++++++++++++++++++++++++++------------------------------
 1 files changed, 40 insertions(+), 45 deletions(-)

febc789: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 15 Dec 2011 10:07:21 -0500
Subject: Remove ChanEval, Add basic escaped-C parsing
Content: No longer require the '$(img:chan)' syntax for channel evaluation. It
should be evaluated implicitly the first time it's used.
```

```
Added basic parsing of escaped-C strings. This allows me to parse out
IDs and present a list of identifiers used by the string. This should
make backend code generation much easier.

 clam/ast.mli          |    5 ++-
 clam/environ.ml       |    3 +-
 clam/parser.mly       |    7 ++---
 clam/printer.ml       |    3 +-
 clam/scanner.mll      |   52 +++++++++++++++++++++++++++++++++++++++++++++----
 clam/tests/sobel.clam |    2 +-
 clam/verifier.ml      |    8 ++++--
 7 files changed, 61 insertions(+), 19 deletions(-)

ec2f068: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 13 Dec 2011 03:03:55 -0500
Subject: C template for use in creating the backend
Content: This is somewhat incomplete, but it's a good start. It should compile
as-is, read in an image and output a grey-scale version of the image.

 clam/libstb/build_template.sh |    8 ++
 clam/libstb/template.c        |  226 ++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 234 insertions(+), 0 deletions(-)

938cfec: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 13 Dec 2011 03:03:05 -0500
Subject: A whole bunch of C machinery to support the backend
Content:
 clam/Makefile                 |    9 +-
 clam/libstb/clam.h            |  324 +++++++++++++++++++++++++++++++++++++++++-
 clam/libstb/stb_image_read.c  |   27 ++++-
 clam/libstb/stb_image_write.c |   36 +++++-
 4 files changed, 390 insertions(+), 6 deletions(-)

5851d63: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 12 Dec 2011 22:15:29 -0500
Subject: Fixed imgwrite error in PNG output
Content:
 clam/libstb/clam.h            |    5 +++++
 clam/libstb/stb_image_read.c  |    2 +-
 clam/libstb/stb_image_write.c |    4 +---
 3 files changed, 7 insertions(+), 4 deletions(-)

357762f: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 12 Dec 2011 10:03:51 -0500
Subject: remove Clamtypes reference
Content:
 clam/parser.mly |    1 -
 1 files changed, 0 insertions(+), 1 deletions(-)

a4e4299: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 9 Dec 2011 20:13:36 -0500
Subject: Beautify the sobel source and fix typo in image name
Content:
 clam/tests/sobel.clam |   17 ++++++++++++++---
 clam/tests/sobel.test |    2 +-
 2 files changed, 15 insertions(+), 4 deletions(-)

1028a27: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 9 Dec 2011 18:23:36 -0500
Subject: Remove clamtypes, move the types to their associated ml files instead.
Content:
 clam/Makefile      |    2 +-
 clam/backend.ml    |    2 +-
 clam/clam.ml       |    3 +--
 clam/clamsys.ml    |    1 -
 clam/clamtypes.ml  |   44 --------------------------------------------
 clam/environ.ml    |   30 +++++++++++++++++++++++++++++-
 clam/parse_util.ml |    4 +++-
 clam/printer.ml    |    3 +--
 clam/verifier.ml   |    1 -
 9 files changed, 36 insertions(+), 54 deletions(-)

9bfca19: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 9 Dec 2011 14:06:54 -0500
Subject: Remove the "-fast" flag so CLAM compiles
Content:
 clam/Makefile |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

db28754: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 8 Dec 2011 23:56:24 -0600
```

```
Subject: removing unnecessary file
Content:
 clam/convop.ml |   39 ---------------------------------------
 1 files changed, 0 insertions(+), 39 deletions(-)

aec2ec5: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 8 Dec 2011 23:54:29 -0600
Subject: Nearly complete verifier!
Content: This version of the verifier completely verifies and successfully
compiles the sobel test. There are a few corner cases let to be checked,
but it's pretty close.

 clam/ast.mli      |   18 ++++---
 clam/backend.ml   |    3 +-
 clam/clamtypes.ml |    7 ++-
 clam/environ.ml   |   10 ++--
 clam/parser.mly   |   15 ++++--
 clam/printer.ml   |   11 ++++-
 clam/verifier.ml  |  135 +++++++++++++++++++++++++++++++++++++++++++-----------------
 7 files changed, 149 insertions(+), 50 deletions(-)

78fec6e: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 8 Dec 2011 23:49:43 -0600
Subject: Add debugging info to Ocaml binary
Content: This allows runtime backtrace generation. You can do this by setting an
environment variable like this:
        export OCAMLRUNPARAM=b
        ./clam
If there is an internal or uncaught exception, you get a nice backtrace!

 clam/Makefile |   13 +++++++------
 1 files changed, 7 insertions(+), 6 deletions(-)

3deac0a: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 7 Dec 2011 19:37:54 -0500
Subject: Unit test documentation update + 1 new simple test
Content:
 clam/tests/README          |   54 +++++++++++++++++++++++++++++++++---------
 clam/tests/_buildup.sh     |    2 +-
 clam/tests/undefined1.clam |   12 +++++++++
 clam/tests/undefined1.test |   15 ++++++++++++
 4 files changed, 70 insertions(+), 13 deletions(-)

928cd97: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 7 Dec 2011 16:34:20 -0500
Subject: adding gitignore
Content:
 clam/.gitignore |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

0168588: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 7 Dec 2011 15:31:52 -0500
Subject: Compile image library, use it in final link
Content:
 clam/Makefile      |   26 ++++++++++++++++++++++----
 clam/clamsys.ml    |    8 +++++---
 clam/libstb/clam.h |    2 ++
 3 files changed, 29 insertions(+), 7 deletions(-)

1c9af9b: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 7 Dec 2011 11:25:43 -0500
Subject: filled in the clam_img_alloc function
Content:
 clam/libstb/clam.h |    8 ++++++++
 1 files changed, 8 insertions(+), 0 deletions(-)

210e12a: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 7 Dec 2011 11:23:21 -0500
Subject: Added imgread/imgwrite interface functions
Content:
 clam/libstb/clam.h            |   21 +++++++++++++++++++++
 clam/libstb/stb_image_read.c  |   34 ++++++++++++++++++++++++++++++++++
 clam/libstb/stb_image_write.c |   37 +++++++++++++++++++++++++++++++++++++
 3 files changed, 92 insertions(+), 0 deletions(-)

fb9d5f3: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 7 Dec 2011 10:55:11 -0500
Subject: Initiali import of stb library for imgread/imgwrite
Content:
 clam/libstb/stb_image_read.c  | 4673 ++++++++++++++++++++++++++++++++++++++++++
 clam/libstb/stb_image_write.c |  511 +++++
 2 files changed, 5184 insertions(+), 0 deletions(-)

e07dcd7: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 7 Dec 2011 10:45:42 -0500
Subject: Tweaks to test framework: give example of invalid compile test
Content:
 clam/tests/_breakdown.sh |    7 ++++-
 clam/tests/_buildup.sh   |   55 +++++++++++++++++++++++++++++++++++++++-----------
 clam/tests/comment1.test |   14 +++--------
 clam/tests/imgcopy.test  |   46 ++++++++++++++++++---------------------------
 clam/tests/invalid1.clam |    1 +
 clam/tests/invalid1.test |   15 ++++++++++++++
 clam/tests/sobel.test    |   38 +++++++++++++++-----------------
 7 files changed, 101 insertions(+), 75 deletions(-)

7d8df66: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 7 Dec 2011 09:23:39 -0500
Subject: Major work on the verifier
Content: It's close to verifying the entire Sobel program! Major missing piece is
adding channels based on the output of convolution.

 clam/ast.mli      |    2 +
 clam/backend.ml   |    6 +-
 clam/clamtypes.ml |   46 +++++++++++++++-
 clam/verifier.ml  |  160 +++++++++++++++++++++++++++++++++++++++++++++++++++++-----------------
 4 files changed, 176 insertions(+), 38 deletions(-)

c80a19a: Robert Martin <rdm2128@columbia.edu>
Date: Tue, 6 Dec 2011 17:22:07 -0500
Subject: Add skeletal code for backend C generation
Content:
 clam/backend.ml |  114 +++++++++++++++++++++++++++++++++++---------------------------
 1 files changed, 62 insertions(+), 52 deletions(-)

0a936df: Robert Martin <rdm2128@columbia.edu>
Date: Tue, 6 Dec 2011 14:32:50 -0500
Subject: Fix error in syntax of imgcopy test, minor cosmetic fix to AST printing, comment-out unused code
      in backend (will be deleted soon)
Content:
 clam/backend.ml         |   67 ++++++++++++++++++++++-------------------------------
 clam/printer.ml         |    3 +-
 clam/tests/imgcopy.clam |    4 +-
 3 files changed, 32 insertions(+), 42 deletions(-)

74f40cd: Robert Martin <rdm2128@columbia.edu>
Date: Tue, 6 Dec 2011 14:22:00 -0500
Subject: Change naming convention for output binaries from *.clam to *.out (because *.clam are the source
      files); Add skeleton for C generation.
Content:
 clam/Makefile   |    2 +-
 clam/backend.ml |   32 +++++++++++++++++++++++++++++++--
 clam/clam.ml    |    4 ++--
 clam/printer.ml |   13 ++++++++-----
 4 files changed, 41 insertions(+), 10 deletions(-)

ed23b4f: Robert Martin <rdm2128@columbia.edu>
Date: Tue, 6 Dec 2011 13:34:54 -0500
Subject: Refactor: create a 'string_of_libf' to translate ImgRead and ImgWrite into strings
Content:
 clam/printer.ml |   27 ++++++---------------
 1 files changed, 6 insertions(+), 21 deletions(-)

51852a1: Robert Martin <rdm2128@columbia.edu>
Date: Tue, 6 Dec 2011 13:32:27 -0500
Subject: Fix formatting bug in --help message
Content:
 clam/clam.ml |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

e172ee1: Robert Martin <rdm2128@columbia.edu>
Date: Tue, 6 Dec 2011 13:28:40 -0500
Subject: Move printing functions to printer.ml. Update other code so it uses the printer.ml stuff.
Content:
 clam/backend.ml   |   20 +++-----------------
 clam/clam.ml      |    2 +-
 clam/clamtypes.ml |   15 ---------------
 clam/environ.ml   |    2 +-
 clam/printer.ml   |   40 ++++++++++++++++++++++++++++++++++++++++
 5 files changed, 45 insertions(+), 34 deletions(-)

0509fdb: Robert Martin <rdm2128@columbia.edu>
```

```
Date: Tue, 6 Dec 2011 13:21:45 -0500
Subject: Split clamtypes into two logical parts: types and environment. Add environ.ml to hold
     environment code.
Content:
 clam/Makefile     |    2 +-
 clam/clamtypes.ml |   57 ----------------------------------------
 clam/environ.ml   |   74 +++++++++++++++++++++++++++++++++++++++++++++++++++++
 clam/verifier.ml  |    1 +
 4 files changed, 76 insertions(+), 58 deletions(-)

69ef8f8: Robert Martin <rdm2128@columbia.edu>
Date: Tue, 6 Dec 2011 12:52:24 -0500
Subject: Add documentation on how we will accept command-line strings into our langauge for imgread and
     imgwrite
Content:
 doc/lrm-expressions.tex |   12 +++++++-----
 1 files changed, 7 insertions(+), 5 deletions(-)

4e85442: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 2 Dec 2011 22:17:05 -0500
Subject: Possible to print AST by adding the "-t" switch. The AST printing still needs some polishing.
Content:
 clam/Makefile   |    2 +-
 clam/clam.ml    |    9 ++++-
 clam/printer.ml |  104 +++++++++++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 112 insertions(+), 3 deletions(-)

b24b521: Robert Martin <rdm2128@columbia.edu>
Date: Fri, 2 Dec 2011 16:57:17 -0500
Subject: Improve the README for the unit tests
Content:
 clam/tests/README |    2 ++
 1 files changed, 2 insertions(+), 0 deletions(-)

e498536: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 1 Dec 2011 15:52:54 -0500
Subject: Redo the test harness, write some simple unit tests
Content:
 clam/Makefile              |    2 +-
 clam/convop.ml             |    4 +-
 clam/tester.sh             |   73 ---------------------------------------------
 clam/tests/README          |   36 ++++++++++++++++++++++
 clam/tests/_breakdown.sh   |   11 +++++++
 clam/tests/_buildup.sh     |   42 ++++++++++++++++++++++++++
 clam/tests/add             |    1 -
 clam/tests/add.res         |   10 ------
 clam/tests/all.test        |   20 ++++++++++++
 clam/tests/assign          |    2 -
 clam/tests/assign.res      |   10 ------
 clam/tests/bad             |    1 -
 clam/tests/bad.res         |   10 ------
 clam/tests/calcdef         |    2 -
 clam/tests/comment         |    1 -
 clam/tests/comment1.clam   |    1 +
 clam/tests/comment1.test   |   22 +++++++++++++
 clam/tests/imgcopy.clam    |    5 +++
 clam/tests/imgcopy.test    |   45 +++++++++++++++++++++++++++
 clam/tests/imgread         |    1 -
 clam/tests/multasn         |    3 --
 clam/tests/multasn.res     |   10 ------
 clam/tests/multistatement  |   10 ------
 clam/tests/sobel           |   31 -------------------
 clam/tests/sobel.clam      |   10 ++++++
 clam/tests/sobel.test      |   36 ++++++++++++++++++++++
 clam/tests/ucla.jpg        |  Bin 0 -> 56626 bytes
 27 files changed, 231 insertions(+), 168 deletions(-)

031df92: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 21 Nov 2011 10:00:55 -0500
Subject: more verifier work... things are in flux :-)
Content:
 clam/clamtypes.ml |    6 ++--
 clam/verifier.ml  |   55 +++++++++++++++++++++++++++++++++++++++++-----------
 2 files changed, 46 insertions(+), 15 deletions(-)

9728676: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 21 Nov 2011 09:15:32 -0500
Subject: Added option to output generated C code
Content:
 clam/clam.ml |   17 +++++++++++++++--
 1 files changed, 15 insertions(+), 2 deletions(-)
```

```
fa18565: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 21 Nov 2011 01:07:06 -0500
Subject: Added the basic verifier framework
Content:
 clam/ast.mli       |    4 ++-
 clam/backend.ml    |    6 ++--
 clam/clam.ml       |    4 +-
 clam/clamtypes.ml  |   88 +++++++++++++++++++++++++++++++++++++++++++-------
 clam/verifier.ml   |   92 ++++++++++++++++++++++++++++++++++++-----------------
 5 files changed, 146 insertions(+), 48 deletions(-)

e480ddf: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 20 Nov 2011 11:20:02 -0500
Subject: don't clear link-order file on every build!
Content:
 clam/Makefile |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

dfce369: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Fri, 18 Nov 2011 10:26:28 -0500
Subject: Added a verifier stub
Content:
 clam/Makefile    |    2 +-
 clam/clam.ml     |    3 ++-
 clam/verifier.ml |   47 +++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 50 insertions(+), 2 deletions(-)

408b378: Yongxu Zhang <yz2419@columbia.edu>
Date: Fri, 18 Nov 2011 01:42:48 -0500
Subject: functions about covolution
Content:
 clam/convop.ml |   39 +++++++++++++++++++++++++++++++++++++++
 1 files changed, 39 insertions(+), 0 deletions(-)

214427a: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 15 Nov 2011 11:02:42 -0500
Subject: count newlines in comments and strings
Content:
 clam/parser.mly   |    4 ----
 clam/scanner.mll  |   15 +++++++++------
 2 files changed, 9 insertions(+), 10 deletions(-)

cc11226: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 15 Nov 2011 02:08:26 -0500
Subject: another test (basically mixed-up sobel)
Content:
 clam/tests/multistatement |   10 ++++++++++
 1 files changed, 10 insertions(+), 0 deletions(-)

36eb200: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 15 Nov 2011 02:03:32 -0500
Subject: Full string/escaped-C/Matrix-definition parsing
Content: So, I stole some string parsing code form the OCaml compiler - very
nice. I leveraged that to get the escaped C parsing. Then I added the
last bit of syntax for now which is the matrix definition.

The resulting binary can now successfully scan/parse the tests/sobel
example CLAM program! Now we need to start getting some smarts in there.

 clam/ast.mli     |   11 +++-
 clam/backend.ml  |   81 ++++++++++++++++---------
 clam/parser.mly  |   38 +++++++++---
 clam/scanner.mll |  175 ++++++++++++++++++++++++++++++++++++++++++++++---------
 4 files changed, 237 insertions(+), 68 deletions(-)

1f8c58c: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 14 Nov 2011 22:46:54 -0500
Subject: added some basic test definitions (no results)
Content:
 clam/tests/calcdef |    2 ++
 clam/tests/comment |    1 +
 clam/tests/imgread |    1 +
 clam/tests/sobel   |   31 +++++++++++++++++++++++++++++++
 4 files changed, 35 insertions(+), 0 deletions(-)

e1ed07d: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 14 Nov 2011 22:43:09 -0500
Subject: Massive commit which adds Syntax error handling
Content: This also beefs up the Makefile to auto-calculate dependencies on the
fly for link and compile time. All you have to do is specify which
```

```
modules to compile :-)

I have brought in a few of the "ExtLib" files from here:
    http://code.google.com/p/ocaml-extlib/
It's a nice library, but I didn't want to force everyone to install it
on their machine.

ISSUE: autodependency calculation doesn't work for incremental builds.
WORK-AROUND: make clean && make
(will be fixed at some point in the future)

 clam/Makefile              |   93 +++++++++---
 clam/ast.mli               |    1 -
 clam/backend.ml            |   12 +-
 clam/clam.ml               |   42 ++++-
 clam/clamtypes.ml          |   18 ++
 clam/extlib/enum.ml        |  376 +++++++++++++++++++++++++++++++++++++++++++++
 clam/extlib/enum.mli       |  201 ++++++++++++++++++++++++
 clam/extlib/extString.ml   |  251 ++++++++++++++++++++++++++++++
 clam/extlib/extString.mli  |  182 ++++++++++++++++++++++
 clam/extlib/std.ml         |  185 ++++++++++++++++++++++
 clam/extlib/std.mli        |   69 ++++++++
 clam/parse_util.ml         |   47 ++++++
 clam/parser.mly            |   10 +-
 clam/scanner.mll           |   16 ++-
 14 files changed, 1460 insertions(+), 43 deletions(-)

85775e9: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 9 Nov 2011 17:51:37 -0500
Subject: Major initial code dump!
Content: WARNING: compiles, but doesn't to anything :-)
This is just to put a stake in the ground which resembles something more
along the lines of what we'll want in our final product.

 clam/Makefile      |    2 +-
 clam/ast.mli       |   36 ++++++++++++++++---
 clam/backend.ml    |   12 ++++++-
 clam/clam.ml       |    2 +-
 clam/clamsys.ml    |    1 +
 clam/clamtypes.ml  |   35 +++++++++++++++++++
 clam/parser.mly    |   97 ++++++++++++++++++++++++++++++++++++++++++++----------
 clam/scanner.mll   |   62 ++++++++++++++++++++++++++++------
 8 files changed, 208 insertions(+), 39 deletions(-)

b480c55: Robert Martin <rdm2128@columbia.edu>
Date: Tue, 1 Nov 2011 19:13:19 -0400
Subject: Work on escaped C expressions, fix keywords, and talk about convolutions
Content:
 doc/lrm-expressions.tex |   44 ++++++++++++++++++++++++++++++++++++++++++--
 doc/lrm-lexical.tex     |    4 ++--
 2 files changed, 44 insertions(+), 4 deletions(-)

79f2c2e: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 1 Nov 2011 00:00:56 -0400
Subject: remove the word annoying
Content:
 doc/lrm-grammar.tex |    3 +--
 1 files changed, 1 insertions(+), 2 deletions(-)

cc0f6c7: Robert Martin <rdm2128@columbia.edu>
Date: Mon, 31 Oct 2011 23:59:00 -0400
Subject: Update some more misc. sections
Content:
 doc/lrm-misc.tex |   17 +++++++++++++----
 1 files changed, 13 insertions(+), 4 deletions(-)

2783ec5: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 23:58:16 -0400
Subject: more expression and object additions
Content:
 doc/lrm-expressions.tex |   51 ++++++++++++++++++++++++++++++++++++++++++++++-
 doc/lrm-misc.tex        |    4 +-
 doc/lrm-objects.tex     |   20 +++++++++++++++++-
 doc/src/sobel.imp       |    2 +-
 4 files changed, 72 insertions(+), 5 deletions(-)

bbed978: Robert Martin <rdm2128@columbia.edu>
Date: Mon, 31 Oct 2011 23:54:01 -0400
Subject: Add statements text and fix some typos
Content:
 doc/lrm-expressions.tex |    2 +-
 doc/lrm-statements.tex  |    7 ++++++-
 2 files changed, 7 insertions(+), 2 deletions(-)

1355974: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 22:17:46 -0400
Subject: update code listing
Content:
 doc/clamlisting.tex |    2 +-
 doc/manual.tex      |    5 +++++
 doc/src/sobel.imp   |   32 ++++++++++++++-----------------
 3 files changed, 20 insertions(+), 19 deletions(-)

677fdf2: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 22:01:31 -0400
Subject: update expression and object discussion (stil incomplete)
Content:
 doc/lrm-expressions.tex |   55 ++++++++++++++++++++++++++++--------------------
 doc/lrm-objects.tex     |   21 +++++++++++++++++--
 2 files changed, 49 insertions(+), 27 deletions(-)

9aca59c: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 21:32:13 -0400
Subject: more nits/updates to lex...
Content:
 doc/clamlisting.tex |    6 +++---
 doc/lrm-lexical.tex |   20 ++++++++++----------
 2 files changed, 13 insertions(+), 13 deletions(-)

8c1966b: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 21:25:11 -0400
Subject: nits and bugfixes to lexigraphical conventions
Content:
 doc/lrm-lexical.tex |   56 +++++++++++++++++++++++++++++++++---------------------
 1 files changed, 33 insertions(+), 23 deletions(-)

5dae2d9: Robert Martin <rdm2128@columbia.edu>
Date: Mon, 31 Oct 2011 21:01:22 -0400
Subject: Update reserved keyword list
Content:
 doc/lrm-lexical.tex |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

f3ee6b1: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 20:52:53 -0400
Subject: split LRM into multiple files
Content:
 doc/lrm-expressions.tex |   95 +++++++++++++++++++++++++++++++++++++++++
 doc/lrm-grammar.tex     |    3 +
 doc/lrm-misc.tex        |   11 ++++
 doc/lrm-objects.tex     |    4 ++
 doc/lrm-statements.tex  |    4 ++
 doc/manual.tex          |  124 ++---------------------------------------------------
 doc/src/sobel.imp       |   11 ++--
 7 files changed, 128 insertions(+), 124 deletions(-)

5b815e7: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 20:47:33 -0400
Subject: split LRM into a couple of files, more basic type work
Content:
 doc/lrm-lexical.tex |   75 ++++++++++++++++++++++++++++++++++++
 doc/lrm-types.tex   |   45 +++++++++++++++++++++
 doc/manual.tex      |  119 +-----------------------------------------------------
 3 files changed, 123 insertions(+), 116 deletions(-)

103803e: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 20:46:51 -0400
Subject: update gitignore
Content:
 .gitignore |    2 ++
 1 files changed, 2 insertions(+), 0 deletions(-)

f7143b3: Robert Martin <rdm2128@columbia.edu>
Date: Mon, 31 Oct 2011 20:33:08 -0400
Subject: Begin work on LRM...
Content:
 doc/manual.tex |   48 ++++++++++++++++++++++++++++++++++++++++++++++++++--
 1 files changed, 46 insertions(+), 2 deletions(-)

789ce29: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 17:42:55 -0400
Subject: initial LRM import: will probably break this up into files for easier collaboration
```

```
Content:
 doc/Makefile    |    1 -
 doc/manual.tex |  202 +++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 doc/paper.bib  |    4 +-
 doc/paper.tex  |   12 ++--
 4 files changed, 209 insertions(+), 10 deletions(-)

3009f6f: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 17:42:29 -0400
Subject: updated example to follow better formulated grammar
Content:
 doc/src/sobel.imp |   32 +++++++++++++++++++-------------------
 1 files changed, 19 insertions(+), 13 deletions(-)

66bfcfa: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 10:39:04 -0400
Subject: check for clam binary before doing regression testing
Content:
 clam/tester.sh |   21 +++++++++++++++++++++
 1 files changed, 21 insertions(+), 0 deletions(-)

a741a43: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 31 Oct 2011 10:31:04 -0400
Subject: no binaries in the git repo please
Content:
 chook/test |  Bin 186429 -> 0 bytes
 1 files changed, 0 insertions(+), 0 deletions(-)

8bf3d0b: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 20 Oct 2011 17:55:58 -0400
Subject: Modify backend to dynamically write C code for the calculator instead of doing calculations at
     compile time
Content:
 clam/backend.ml |   17 +++++++++--------
 1 files changed, 9 insertions(+), 8 deletions(-)

e48a604: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 20 Oct 2011 15:49:41 -0400
Subject: Simple as possible program that hooks C functions into ocaml, in case we need to do this
Content:
 chook/Makefile   |   17 +++++++++++++++++
 chook/dummy.mli  |    3 +++
 chook/test       |  Bin 0 -> 186429 bytes
 chook/test.ml    |    7 +++++++
 chook/wrap.c     |   22 ++++++++++++++++++++++
 5 files changed, 49 insertions(+), 0 deletions(-)

2db1c23: Robert Martin <rdm2128@columbia.edu>
Date: Thu, 20 Oct 2011 12:57:39 -0400
Subject: Add removal of 'clam' binaries to 'make clean'
Content:
 clam/Makefile |    4 ++++
 1 files changed, 4 insertions(+), 0 deletions(-)

a3fdf74: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Thu, 20 Oct 2011 11:48:14 -0400
Subject: beautify tester + add compile output checking
Content:
 clam/Makefile           |    3 ++
 clam/tester.sh          |   59 +++++++++++++++++++++++++++++++++--------------------
 clam/tests/add.res      |    9 +++++++
 clam/tests/assign.res   |   11 ++++++++-
 clam/tests/bad.res      |   11 ++++++++-
 clam/tests/multasn.res |    9 +++++++
 6 files changed, 82 insertions(+), 20 deletions(-)

26955b2: Kevin Sun <kfs2110@columbia.edu>
Date: Tue, 18 Oct 2011 20:31:37 -0400
Subject: Added a basic shell script for testing, and some examples
Content: Files to be tested are in the folder clam/tests, and each has a corresponding
"result file" ending in .res -- so "add" goes with "add.res" etc.
tester.sh just goes through all the test files, tries to compile and run them,
and reports any problems - i.e. failure to compile, or unexpected output etc.

(The examples are for the calculator and not actual CLAM, so I expect we'll scrap them later.)

 clam/scanner.mll        |    2 +-
 clam/tester.sh          |   29 +++++++++++++++++++++++++++++
 clam/tests/add          |    1 +
 clam/tests/add.res      |    1 +
 clam/tests/assign       |    2 ++
 clam/tests/assign.res  |    1 +
 clam/tests/bad         |    1 +
 clam/tests/bad.res     |    1 +
 clam/tests/multasn     |    3 +++
 clam/tests/multasn.res |    1 +
 10 files changed, 41 insertions(+), 1 deletions(-)

4ec105e: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Tue, 18 Oct 2011 00:02:54 -0400
Subject: Major framework update
Content: CLAM can now accept input on stdin, "compile"(currently:interpret) the
input, generate a C file, invoke gcc to compile the C file, and then
link a machine executable binary!

Try it out with:
    echo "23 + 23 - 4" | ./clam
    ./bin.clam

 clam/Makefile    |   56 ++++++++++++++++++++++++++++++++++++++------------------
 clam/backend.ml  |   50 +++++++++++++++++++++++++++++++++++++++++++++++++++++
 clam/clam.ml     |   47 ++++++++++++++++++++-----------------------------------
 clam/clamsys.ml  |   61 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 clam/parser.mly  |    9 ++++++--
 clam/scanner.mll |   11 ++++++---
 6 files changed, 171 insertions(+), 63 deletions(-)

d289ab7: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 17 Oct 2011 18:00:50 -0400
Subject: updated gitignore
Content:
 .gitignore |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

fde0600: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 16 Oct 2011 20:00:18 -0400
Subject: cleanall -> distclean
Content:
 doc/Makefile |    4 ++--
 1 files changed, 2 insertions(+), 2 deletions(-)

1e03201: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 16 Oct 2011 19:58:57 -0400
Subject: default to LRM compilation: edit manual.tex
Content:
 doc/Makefile        |    4 ++--
 doc/clamlisting.tex |   35 +++++++++++++++++++++++++++++++++++
 doc/manual.tex      |    3 +--
 doc/paper.tex       |    5 +++--
 doc/proposal.tex    |    8 --------
 5 files changed, 41 insertions(+), 14 deletions(-)

583ad60: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Sun, 16 Oct 2011 19:40:57 -0400
Subject: initial import of calculator example
Content:
 .gitignore       |    3 ++
 clam/Makefile    |   81 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 clam/ast.mli     |   19 ++++++++++++
 clam/clam.ml     |   51 +++++++++++++++++++++++++++++++++++
 clam/parser.mly  |   35 +++++++++++++++++++++++
 clam/scanner.mll |   23 +++++++++++++++
 src/README       |    1 -
 7 files changed, 212 insertions(+), 1 deletions(-)

352f7b0: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 28 Sep 2011 15:43:13 -0400
Subject: nits
Content:
 doc/proposal.tex  |   10 +++-------
 doc/src/sobel.imp |    2 +-
 2 files changed, 4 insertions(+), 8 deletions(-)

637fc78: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 28 Sep 2011 15:10:45 -0400
Subject: Initial language proposal document
Content:
 doc/proposal.tex  |  150 +++++++++++++++++++++++++++++++++++++++++++++------------------------
 doc/src/sobel.imp |   24 +++------
 2 files changed, 104 insertions(+), 70 deletions(-)

0b1315e: Jeremy C. Andrus <jeremya@cs.columbia.edu>
```

```
Date: Wed, 28 Sep 2011 15:10:25 -0400
Subject: updated formatting: changed name to CLAM
Content:
 doc/paper.tex |   14 +++++++++-----
 1 files changed, 9 insertions(+), 5 deletions(-)

7d7a16d: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 28 Sep 2011 15:09:12 -0400
Subject: removed text from intro
Content:
 doc/intro.tex |    7 -------
 1 files changed, 0 insertions(+), 7 deletions(-)

179f463: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Wed, 28 Sep 2011 15:07:34 -0400
Subject: updated gitignore
Content:
 .gitignore |    3 ++-
 1 files changed, 2 insertions(+), 1 deletions(-)

2f85eac: Robert Martin <rdm2128@columbia.edu>
Date: Wed, 28 Sep 2011 11:15:15 -0400
Subject: Add language basic language syntax concepts and simple example problem to the proposal
Content:
 doc/intro.tex    |    2 +-
 doc/proposal.tex |   52 +++++++++++++++++++++++++++++++++++++++++++++++++++-
 2 files changed, 52 insertions(+), 2 deletions(-)

00e912e: Yongxu Zhang <yz2419@columbia.edu>
Date: Wed, 28 Sep 2011 02:38:45 -0400
Subject: Edited intro.tex
Content: Wrote brief description of language, and suggested some possible names.

 doc/intro.tex |    8 ++++++++
 1 files changed, 8 insertions(+), 0 deletions(-)

5dc010d: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 26 Sep 2011 19:58:56 -0400
Subject: removed unnecessary file
Content:
 doc/acm_proc_onecol.cls | 1629 -----------------------------------------------
 1 files changed, 0 insertions(+), 1629 deletions(-)

c022caf: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 26 Sep 2011 19:56:06 -0400
Subject: removed old code sample
Content:
 doc/src/rdm_sample_code.txt |  107 -------------------------------------------
 1 files changed, 0 insertions(+), 107 deletions(-)

5c2c614: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 26 Sep 2011 19:43:27 -0400
Subject: Updated LaTeX source - edit proposal.tex for proposal (type make to build)
Content:
 doc/Makefile     |   17 ++++++++++++++---
 doc/paper.tex    |   42 +++++-------------------------------------
 doc/proposal.tex |    3 ++-
 3 files changed, 21 insertions(+), 41 deletions(-)

5f3c12c: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 26 Sep 2011 19:42:51 -0400
Subject: added gitignore file
Content:
 .gitignore |    9 +++++++++
 1 files changed, 9 insertions(+), 0 deletions(-)

3c24a92: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 26 Sep 2011 19:41:38 -0400
Subject: updated source example
Content:
 doc/src/sobel.imp |   36 ++++++++++++++++++++++++------------
 1 files changed, 24 insertions(+), 12 deletions(-)

dd77f95: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 26 Sep 2011 16:10:05 -0400
Subject: renamed and moved sample code file
Content:
 doc/sample_code.txt         |  107 -------------------------------------------
 doc/src/rdm_sample_code.txt |  107 +++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 107 insertions(+), 107 deletions(-)
```

```
91d43a6: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 26 Sep 2011 16:09:15 -0400
Subject: initial latex document import
Content:
 doc/Makefile            |  117 ++++
 doc/README              |    1 -
 doc/acm_proc_onecol.cls | 1629 +++++++++++++++++++++++++++++++++++++++++++++++
 doc/appendix.tex        |    2 +
 doc/design.tex          |    2 +
 doc/intro.tex           |    3 +
 doc/lessons.tex         |    2 +
 doc/manual.tex          |    2 +
 doc/paper.bib           |   10 +
 doc/paper.tex           |  132 ++++
 doc/paper.tex.latexmain |    1 +
 doc/plan.tex            |    2 +
 doc/proposal.tex        |   18 +
 doc/src/sobel.imp       |   24 +
 doc/testing.tex         |    1 +
 doc/tutorial.tex        |    3 +
 16 files changed, 1948 insertions(+), 1 deletions(-)

038d750: Robert Martin <rdm2128@columbia.edu>
Date: Mon, 26 Sep 2011 15:19:10 -0400
Subject: Add some hypothetical sample code
Content:
 doc/sample_code.txt |  107 +++++++++++++++++++++++++++++++++++++++++++++++++++
 1 files changed, 107 insertions(+), 0 deletions(-)

659829a: Jeremy C. Andrus <jeremya@cs.columbia.edu>
Date: Mon, 19 Sep 2011 23:43:39 -0400
Subject: Initial import
Content:
 doc/README |    1 +
 src/README |    1 +
 2 files changed, 2 insertions(+), 0 deletions(-)
```

Bibliography

[1] S. Barrett. Public domain JPEG and PNG decompression, BMP, TGA, PSD loading. `http://nothings.org/stb_image.c`.

[2] S. Barrett. Public domain very-limited image writers. `http://nothings.org/stb_image_write.h`.

[3] N. Cannasse, B. Hurt, Y. Yoriyuki, and J. Hellsten. ocaml–extlib – OCaml ExtLib – Extended Standard Library for Objective Caml – Google Project Hosting. `http://code.google.com/p/ocaml-extlib/`.

[4] B. W. Kernighan and D. Ritchie. *The C Programming Language, Second Edition.* Prentice-Hall, 1988.

[5] L. Torvalds, J. Hamano, and Git Community. Git – Fast Version Control System. `http://git-scm.com/`.

[6] Wikipedia Community. Sobel operator – Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Sobel_operator`.