YOLT Programming Language Final Report

Yuan Zheng
Omar Ahmed
Lukas Dudkowski
Takahiro Mark Kuba

May 2003
CS4115

Contents

1 Dedications

Lukas would like to dedicate this final report to this kitten, who kept us all sane.

With special thanks to Arizona Iced Tea Co., Hamilton Deli, and those CLIC lab chairs with our very own a** grooves.

2 Introduction

YOLT is a "web page" language whose basic idea is to download HTML from the web and perform certain transforms and operations on the data in order to generate necessary information in a number of desirable formats. Data can be aggregated from a number of different web sites, and could be transformed or distilled for other purposes. YOLT is a cross-platform language with natural language processing uses. It overcomes the major limitations of HTML and RSS feeds. It is network aware and cross-platform, with rich regular expression support.

2.1 Background

2.1.1 HTML Limitations

Currently, HTML is not used as a document markup language based on textual organization, as originally intended. Instead, it is a graphical layout markup language used for displaying web pages. This makes parsing useful information out of HTML to be difficult, as the content and display information are not easily separated. One problem with web page layout is that the content server has complete control over how the information is to be displayed. User defined style sheets are supposed to give the user limited control over the display of information. This runs into problems, however, when one wants to view web pages in ways the author did not originally intend. For example, reformatting web pages for disability access, or also reformatting web pages for limited displays (such as personal digital assistants or cell phones) are cases the author may not have originally addressed.

2.1.2 RSS Feeds Limited

RSS, or Rich Site Summary format is one approach to solving the problem with HTML. RSS feeds are provided by web site owners normally, and the XML format allows people to display the information as they choose to. However, the main limitation of RSS feeds is that few people provide public RSS feeds for use, thus limiting the utility of the technology.

2.2 Goals, Features

2.2.1 Natural Language Processing Uses

One use for YOLT is for natural language processing uses. Often times, the raw content is required by natural language applications. However, due to the problems and limitations of HTML, often times, the layout information and the content information cannot easily be separated. This presents an entire host of problems in and of itself. What YOLT can do in this domain is solve this problem for natural language processors and allow them to focus more on interpreting and working with the raw data instead of working to try and separate content from layout information.

2.2.2 Accessibility Uses

YOLT can similarly be used for those who are visually impaired. Extracting content and passing it to a speech synthesis program could allow one to hear the contents of CNN.com. Current screenreaders are very limited in their HTML interpretive capabilities. Use of YOLT could allow one to have the content read aloud without the "noise" of the display information.

2.2.3 Ubiquitous Computing Applications

In the same manner, this ability to abstract information out of web sites could be very useful for limited display devices. For example, many web sites are designed for computer monitors at a high resolution, supporting thousands or millions of colors. This fails on a monochrome low resolution cell phone display or personal digital assistant. Previously, some web site owners have tried to provide low-res versions of web sites for these applications, but the use is not widespread enough to be of any real use.

2.2.4 Network Aware

Another feature of YOLT will be the network-aware nature of the language. Since most of the applications are to be done on web pages, the HTTP protocol will definitely be supported. This will allow one to download web pages and perform operations on them without resorting to outside libraries or socket programming. The process will be simplified greatly for the benefit of utility and ease of use of the language.

2.2.5 Cross-platform

YOLT will also be a cross-platform language. As the web itself is platform-neutral, it only makes sense that YOLT also be platform-neutral. On compilation, YOLT will generate Perl code, which is itself platform neutral. Therefore, the advanced features of YOLT will be readily available to different users of different platforms, so that anyone can take full advantage of the web's ubiquitous presence.

2.2.6 Regular Expressions

YOLT will employ a powerful regular expression engine to allow one to easily parse web pages. This is necessary for the matching that needs to be done in order to extract appropriate information from the source material. Regular expression matching will greatly assist those who would like to use natural language capabilities with the web.

2.2.7 Simplicity

YOLT will be a fairly simple language, mostly focusing on text and string manipulation. The model of computation will be like Perl, a simple scripting language based on pattern matching.

2.3 Sample Code

```
begin

#download web page into foo variable
$foo := "http://www.nytimes.com/";

#regular expr to get stories between links
$story_tags="<a href=\"(.*)\">.*</a>";

# pattern match on web page, store in variable bar
$bar = $story_tags @ $foo;

# for each pattern match
foreach $story in $bar {

$baz = $headline_tags @ $story;
# print out this headline
echo $baz;
}
end
```

3 Language Tutorial

3.1 A Basic Program

Here is the basic YOLT program that we'll use to get started.

```
begin
# Program to do the obvious
        echo "Hello world.";            # Print a message
end
```

Each of the parts will be discussed in turn.

3.1.1 The first line

Every YOLT program must begin with a "begin" statement.

3.1.2 Comments and statements

Comments can be inserted into a program with the # symbol, and anything from the # to the end of the line is ignored. The only way to stretch comments over several lines is to use a # on each line. Everything else is a YOLT statement which must end with a semicolon, like the echo line above.

3.1.3 Simple printing

The *echo* function outputs some information. In the above case it prints out the literal string "Hello world." and of course the statement ends with a semicolon.

3.1.4 The last line

Every YOLT program must end with the "end" statement.

3.2 Running the program

"yolt" is a simple script that launches the YOLT interpreter. Open the "yolt" script and point the first line at the location of bash on your particular machine. For example:

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

Type in the example program using a text editor, and save it. Now to run the program just type the following at the prompt.

```
./yolt programname.y
```

You should get the expected output though it isn't very pretty.

3.2.1 Output redirection

Meant as an "information scraping language" YOLT's functionality is ideally suited towards generating html output files for example, containing the desired information (news headlines, comics, weather, etc). Towards this end, the yolt script will accept a second command line argument, the output file. To put the above "Hello World." output into a file *greeting.file*, we would simply call the yolt script with two command line arguments as below:

```
./yolt programname.y outputfile
```

Where *outputfile* would be replaced by *greeting.file*

3.3 Scalar variables

The most basic kind of variable in Yolt is the scalar variable. Scalar variables hold both strings and numbers, and are remarkable in that strings and numbers are completely interchangeable. For example, the statement

```
$priority = 9;
```

sets the scalar variable $priority to 9, but you can also assign a string to exactly the same variable:

```
$priority = "high";
```

Yolt also accepts numbers as strings, like this:

```
$priority = "9";
$default = "0009";
```

and can still cope with arithmetic and other operations quite happily.
In general variable names consists of numbers, letters and underscores. Also, Yolt is case sensitive, so $a and $A are different.

3.3.1 Operations and assignment

YOLT uses the following arithmetic operators:

```
$a = 1 + 2;                    # Add 1 and 2 and store in $a
$a = 3 - 4;                    # Subtract 4 from 3 and store in $a
$a = 5 * 6;                    # Multiply 5 and 6
$a = 7 / 8;                    # Divide 7 by 8 to give 0.875
$a = 5 % 2;                    # Remainder of 5 divided by 2
```

3.4 Array variables

Slightly more interesting are array variables which are lists of scalars (i.e. numbers and strings). Array variables have the same format as scalar variables and are not really differentiated in YOLT.

```
$food  = {"apples", "pears", "eels"};
$music = {"whistle", "flute"};
```

assigns a three element list to the array variable $food and a two element list to the array variable $music. The array is accessed by using indices starting from 0, and square brackets are used to specify the index. The expression

```
$food[2]
```

returns *eels*.

3.4.1 Array assignments

Arrays can be assigned values by either initializing them to values using the curly braces, assigning them to expressions that evaluate to arrays, or individually accessing elements in the array and assigning them values as though they were scalars.

```
# initializing an array to values
$beatles = {‘‘John’’,’’Paul’’,’’George’’,’’Ringo’’};

# expressions which evaluate to arrays
$classes = call new_array(‘‘fall’’,2003);

# individually assigning values to elements in the array
$beatles[4] = ‘‘Oscar’’;
```

3.4.2 Displaying arrays

To iterate through the contents of an array or other list-like structure YOLT provides the ”foreach” statement. For example, if we have an array $students, with contents of ”Matthew”, ”Mark” , ”Luke” , and ”John” we can print the contents of this array with the following use of the ”foreach” statement

```
# ... some code ...

foreach $x in $students {
        echo $x;
        echo "\n";
}

# ... rest of code ...
```

The actions to be performed each time are enclosed in a block of curly braces. The first time through the block $x is assigned the value of the first item in the array $students. Next time it is assigned the value of the second item, and so until the end. If $students is empty to start with then the block of statements is never executed. The code above will print the contents of the array as:

```
Matthew
Mark
Luke
John
```

3.5 Control structures

3.5.1 foreach

see *Displaying Arrays* above.

3.5.2 testing

The next few structures rely on a test being true or false. In YOLT any non-zero number and non-empty string is counted as true. The number zero, zero by itself in a string, and the empty string are counted as false. Here are some tests on numbers and strings.

```
$a == $b                                # Is $a numerically equal to $b?
$a != $b                                # Is $a numerically unequal to $b?
$a <= $b                                # Is $a less than or equal to $b?
$a >= $b                                # Is $a greater than or equal to $b?
```

You can also use logical and, or and not:

```
($a && $b)              # Is $a and $b true?
($a || $b)              # Is either $a or $b true?
!($a)                   # is $a false?
```

3.5.3 while

While loops are handled in an expected manner. The format is while (conditional) statements, where the conditional is tested at the beginning of the loop. If the conditional is false, then the contents of the while loop are never executed.

3.6 Conditionals

YOLT also allows if/then/else statements. These are of the following form:

```
if ($a)
{
        echo "The string is not empty\n";
}
else
{
        echo "The string is empty\n";
}
```

For this, remember that an empty string is considered to be false. It will also give an ”empty” result if $a is the string 0.
It is also possible to include more alternatives in a conditional statement:

```
if ($a) # The ! is the not operator
{
        echo "The string is not empty\n";
}
else if ($a == "John")           # The empty test has failed, so it is John?
{
        echo "Hello John!";
}
```

3.7 String matching

One of the more powerful features of YOLT is its string matching capabilities and regular expression support. This is very useful for web scraping, as you usually want to match against a pattern of markup language in order to pull out the information that you are interested in.

3.7.1 Regular expressions

Regular expression support is provided by using the '@' operator. See the section on "Special Operators" for more information on how to use it.

3.8 Functions

Like any good programming language YOLT allows the user to define their own functions. They may be placed anywhere in your program but it's probably best to put them all at the beginning or all at the end. All functions must be declared at the beginning of the program, after the "begin" statement but before any other statements. A subroutine has the form

```
declare myfunction();

# ... program code ...

call myfunction();

# ... more code ...

function myfunction()
{
        echo "Not a very interesting routine\n";
        echo "This does the same thing every time\n";
}
```

The following will call the function "myfunction" with no args.

```
call myfunction(); # Call the subroutine
```

3.8.1 Parameters

The above example has no parameters passed to it. If we wanted to declare a function "sum" for example, that takes two args and prints their sum we would use the following declaration:

```
declare sum($a, $b);

# program code.......
```

```
function sum($a, $b)
{
        echo "Sum is: ";
        echo $a+$b;
        echo "\n";
}

sum(3, 4);              # prints "Sum is: 7"
sum(2, 2);              # prints "Sum is: 4"
```

3.8.2 Returning values

If we wanted to return the sum of those two variables above, we would declare the ”sum” function as above:

```
declare sum($a, $b);
```

However, within the function we would use the ”return” statement

```
function sum{$a, $b)
{
        return $a+$b;
}
```

To implement the printing sum functionality we could then call sum, and print its return value, or alternatively assign it to a variable.

```
echo call sum(2, 2);            # prints 4

# or alternatively

$c = call sum(2, 2);            # $c gets 4
echo $c; # prints 4
```

YOLT will ignore the return result if the user is calling a function that returns a variable without an explicit assignment, for example:

```
# program code....

call sum(2, 2);

# program code....
```

This will produce no output, and YOLT will ignore the returned sum result.

3.9 Special Operators

3.9.1 The := operator

The web get operator will download the web page associated by the URL the string points to, and store the HTML into the identifier. Upon error, a negative number of the error (i.e. -404 is assigned if the server returns a Web Page Not Found error)
The "gets" operator is one of the most useful features of the YOLT language. For example, to print the html contents of www.yahoo.com, a user would enter the following YOLT program named "yahoo.y":

```
begin

$page:="http://www.yahoo.com"; #$page "gets" yahoo.com

foreach $line in $page{ #for every line in the page do

        echo $line; #print the line's contents

}
end
```

This will put the html contents of the website onto stdout, if we wanted to say, put the contents of the site into an html file in our local directory, we would run the yolt script with in the following manner:

```
./yolt yahoo.y yahoo.html
```

This would put the html contents of the website (at the URL) into the local file "yahoo.html."

3.9.2 The @ operator

The '@' symbol is used when matching with regular expressions. The format of the expression on the left hand side of the @ symbol is a regular expression with parentheses around the data to be retained. The expression on the right hand side of the @ symbol is the text against which you are matching. For instance, let's take the above program in which we downloaded the html contents of yahoo.com. If instead of getting the whole html contents we wanted to "scrape" out only the names of any hyperlinks on the yahoo.com site, we would alter the above program only slightly.

```
begin

$page:="http://www.yahoo.com"; #same as before

$tags = "<a href=.*>(.*)</a>"; #regular expression, will match for hyperlink names
#the item of interest is in ()'s, to print out only the
```

```
#hyperlinks we would enclose the whole tag in ()'s
#like:  "(<a href.... </a>)"

$links = $tags @ $page; # the "at" operator extracts any matching lines with
#the tags above, and stores that in the $links array

foreach $line in $links{ #here we change to walk the $links, not $page
#array, so print out everything in the $links array
#for clarity, also put a newline.
        echo $line;
        echo "\n";
}
```

The resulting output is the list of the names of the links on the main site of yahoo.com. Here is a sample of the output which this code would actually generate...

```
.
.
.
.
Send a free e-card to Mom
Y! Photos: posting pictures is free
Auctions
Autos
Classifieds
Real Estate

Travel
HotJobs
Maps
People Search
Personals
Yellow Pages
Chat
GeoCities

Groups
Mail
Messenger
Mobile
Addresses
Briefcase
Calendar
My Yahoo!
PayDirect
```

```
Games
Horoscopes
Kids
Movies
Music
Platinum
TV
Finance
Health
News
Sports
Weather
More Yahoo!...
.
.
.
.
```

also see LRM for further description

3.10 embedded Perl

The inclusion of Perl blocks was done to facilitate YOLT use by "power-users" who may wish to access the powerful regular-expression handling of Perl, as well as other features of that language like error-checking through eval, or in specific cases, file handling.
To include an embedded Perl block, you have to enclose it within &{ }&

```
begin
# ... YOLT code ...

&{
perl block
}&

# ... YOLT code.....
end
```

4 Language Reference Manual

4.1 Lexical Conventions

4.1.1 Line Terminators

Input is divided into lines by line terminators. Lines are terminated by the standard ASCII carriage-return/linefeed. To support the three different newline standards, either carriage-return or linefeed used alone can function as a line terminator.

4.1.2 Comments

Only single-line comments are permissible in YOLT. Comments will begin with a '#' and will continue until a line terminator is reached. A '#' inside an existing comment will have no meaning.

4.1.3 Whitespace

Whitespace in YOLT consists of the following: indentations, spaces, tabs, and line terminators. Whitespace will be ignored, except when used to terminate a comment (a line terminator).

4.1.4 Tokens

Tokens are subdivided into 4 classifications: identifiers, keywords, constants, and separators. Tokens must be separated by whitespace. Much like other existing languages, token formation in YOLT is greedy; i.e. a token is the longest string of input characters that can constitute a token.

4.1.5 Identifiers

Identifiers in YOLT must begin with a '$' symbol. This distinguishes them from keywords and other language constructs. Therefore, an identifier in YOLT could technically be the same word as a keyword, only with a '$' in front of it. After the '$' symbol can follow any sequence of numbers, digits, and the underscore character. In fact, an identifier can consist solely of the '$' and a sequence of underscores, or a sequence of numbers. If at the end of the identifier appear square brackets at the first use of the identifier ([]), the identifier points to an array. Multidimensional arrays are allowed with multiple sequences of brackets, i.e. $myArray[5][5]

*Identifier -> $(letter|digit|underscore)**

4.1.6 Keywords

The following sequences of characters are reserved for use as keywords:

```
 foreach
in
while
if
else
echo
function
include
```

```
return
declare
begin
end
```

4.1.7 Integer Constants

Constants are sequences of ASCII digits which represent an integer literal. Constants lie in the range of the Java primitive type "int"

IntConstant -> digit+

4.1.8 String Constants

Any String in YOLT is a sequence of characters, digits, or symbols surrounded by double quotations *StrConstant -> "*"*

To embed a quotation ' " ' in a String, the C-style use of the backslash as an "escape character" will be implemented. i.e. if a String is to contain the following: *he said "hello"* it will look as follows:

```
"he said \"hello\""
```

The following characters can be escaped by placing a \ before them: $,,",n,r,t,b,f,',.,(,),?

4.1.9 Separators

The following ASCII characters are separators:
{ } [ ]

4.2 Expressions

4.2.1 Primary Expressions

Identifiers, string constants, and integer constants are primary expressions.

4.2.2 Unary Operators

!expression
Negates the expression.

4.2.3 Multiplicative Operators

*expression * expression*
First expression times the second expression
expression / expression
First expression divided by the second expression

4.2.4 Boolean Operators

boolean or
expression || expression
boolean and
expression && expression
on divided by the second expression
expression % expression
Returns the modulus. The remainder from the division of the first expression by the second expression.

4.2.5 Additive Operators

expression + expression
expression - expression

4.2.6 Relational Operators

expression >= expression
expression > expression
expression < expression
expression <= expression

4.2.7 Equality Operators

expression == expression
expression != expression

4.2.8 Boolean Operators

boolean or
expression || expression
boolean and
expression && expression

4.2.9 Concatenation Operator

string . string Concatenates two strings into one

4.2.10 identifier := string

Web get operator will download the web page associated by the URL that the string points to, and store the HTML into the identifier. Upon error, a negative number of the error (i.e. -404 is assigned if the server returns a Web Page Not Found error)

4.2.11 identifier = expression

assignment operator

4.2.12 expression @ expression

The '@' symbol is used when matching with regular expressions. The format of the expression on the left hand side of the @ symbol is a regular expression with parentheses around the data to be retained. The expression on the right hand side of the @ symbol is the text against which you are matching.
$x = "(needle)" @ $haystack;

4.3 Declarations

The only declarations in YOLT are function declarations. Variables are implicitly declared upon use.

4.4 Function declaration

declare function-name(identifier-list);
Functions must be declared in the beginning of the source file, after the begin keyword but before any other code. Functions are defined, however, in a different location.

4.4.1 Function definition

function| function-name(identifier-list) statement
Where function-name follow the same rules as identifiers, except without the beginning '$' character. identifier-list is a comma-separated list of identifiers. The statement must include a return statement inside of it to return program control to the caller.

4.5 Statements

4.5.1 Expression statement

Most expressions will be of the form
expression;
and will usually be assignments or function calls.

4.5.2 Compound Statement

Compound statements are provided so that several statements can be included where one statement is expected.
compound-statement -> { (statement-list)* }
— (statement)

4.5.3 Conditional Statement

Conditionals are done with if statements, in the forms if (expression) statement or if (expression) statement else statement In the first example, the statement is executed if the expression is true. In the second example, the first statement is executed if the expression is true, and the second statement is executed otherwise. In cases of the "dangling else" problem, the else is connected to the nearest unmatched if.

4.5.4 while statement

The while statement is in the form while (expression) statement And is used for program looping. The check is done before execution of the statement, and will be repeated as long as the test expression is true.

4.5.5 foreach statement

The foreach statement is in the form *foreach identifier1 in identifier2 statement*

This is used primarily to loop through array elements, where identifier1 is used inside the statement to refer to one particular element of the identifier2 array.

4.5.6 return statement

The return statement is used to return to the caller. This is only to be used for defined functions

return; # this will return control to the function caller
return (expression); # this will return control, and also return a value

4.5.7 null statement

A null statement has the form *;*

4.5.8 echo statement

echo expression
Used to print to stdout. No other forms of output are supported.

4.6 Scope Rules

4.6.1 Lexical scope

YOLT is a language with open scope and only a global namespace.

4.7 CFG

see Appendix for CFG

5 Project Plan

5.1 Planning, Specification

Planning and specification occurred in February and March in several meetings, with the TA and also with just the group. See the Project Log for specific dates of when decisions were made and milestones achieved.

5.2 Project Timeline

Here is a timeline of major project milestones:

11	Feb	Finalization of language concept
15	Feb	Whitepaper draft
18	Feb	Whitepaper due
12	Mar	LRM draft
27	Mar	LRM due
8	Apr	Lexer/Parser
15	Apr	Code generator/semantic checking
22	Apr	Integration
6	May	Feature freeze, final report draft
13	May	Final report due

5.3 Roles and Responsibilities

Our group decided to split the responsibilities of implementing YOLT along four major divisions; Parser and Lexer, Semantic Checking, Code Generation, and Documentation and Testing.

T. Mark Kuba Lexer, Parser, Example Programs

My primary responsibilities for the project were the ANTLR grammars for the lexer and the parser, and also writing example YOLT programs. To generate the grammar, I first started by transferring the CFG that was submitted for the original Language Reference Manual. This was then adapted to remove left-recursion as well as non-determinism errors in ANTLR. I tested for bugs in the grammar as I was writing it, further bugs were discovered upon the formal testing in isolation, and even more were discovered during integration. My other responsibility was writing example YOLT programs to demonstrate the utility of the language. This was a good way of uncovering obscure bugs in the integration of all of the different parts of the project. This was also a very exciting part of the project, to see the various components of the language integrated and working together. Seeing some of the first information scrapes was a great reward for all of the work that was put into the project.

Omar Ahmed Semantic Checking

Originally, I had decided to do semantic checking with some basic types. Our language is loosely typed, so I assumed that I would need some basic type checking (i.e. to make sure that things are assigned to the right type). However, after I had written over 1,000 lines of code, we realized that the type checking was unnecessary. I got rid of all type checking from my code, leaving it with minimum functionality–make sure that all variables that are used are declared, and making sure that functions are declared, and making sure that everything is formed correctly, so that things are not out of place.

Yuan Zheng Code Generation

My responsibility in YOLT is generating Perl code using the sementically checked AST tree built by the parser. In order to generate code, I walked the whole tree. According to the specific parent node information, I either go down to the children node or generate perl code. This process is repeated until every single AST node is reached. I used incremental developing and testing during my code generation. Generate the simplest code first, then add more complicated operations after successfully tested the early code. Regression testing is implemented in my code developing. All the old test cases are saved and retested in the next development phase.

Lukas Dudkowski Documentation and Testing

My responsibility was the generating this documentation as well as testing and integration of the components that make up the YOLT interpreter. This involved testing during the development phase of each component by working with the programmer and incremental test of implemented features and functionality, as well as interface testing once major components were implemented. I wrote and ran test cases on the lexer/parser, semantic checker, and code generator throughout their development cycles providing feedback to the developers. Along with Mark, I wrote sample YOLT programs to test functionality of the language during various integration phases, as well as making sure all components were behaving in the expected manner given known-good inputs.

5.4 Tools and Languages Used

Languages	Java, Perl, Bash Scripts
Tools	NetBeans, Antlr (Lexer/Parser), Bash (test scripting), Kile (LaTeX frontend), Emacs

5.5 Project Log

Below is a project log of important milestones achieved.

9	Feb	Language concept finalized
16	Feb	Whitepaper draft
18	Feb	Whitepaper submitted
27	Feb	Discuss whitepaper with TA
9	Mar	LRM draft
24	Mar	Discuss LRM with TA
27	Mar	LRM submitted
6	Apr	YOLT implementation planning meeting
12	Apr	Lexer/Parser
30	Apr	Code generation
3	May	Semantic checking & integration/testing
7	May	Draft of final report
13	May	Final report submitted

6 Architectural Design

YOLT is an interpreted language whose main purpose is simple information gathering, ”scraping.” Several architectural choices were made along the development cycle with the goal of a simple, easily maintainable, yet capable language. ”Power-user” functionality was left out of YOLT itself, though the option for error checking, file handling, system-call access, and other ”power-user” features has been made possible through the ability to embed a Perl block within a YOLT program.
The target language of YOLT is Perl. As a simple, network-aware language, YOLT aims to simplify the action of information gathering over more-capable, powerful, and complex languages such as Perl. In this regard, the simplicity of a YOLT program wins over that of Perl in tasks such as scraping comics from various sources and generating a ”daily comics” page. When viewed side-by-side, YOLT programs are smaller, more intuitive, and in this set of tasks, as powerful as equivalent Perl programs.

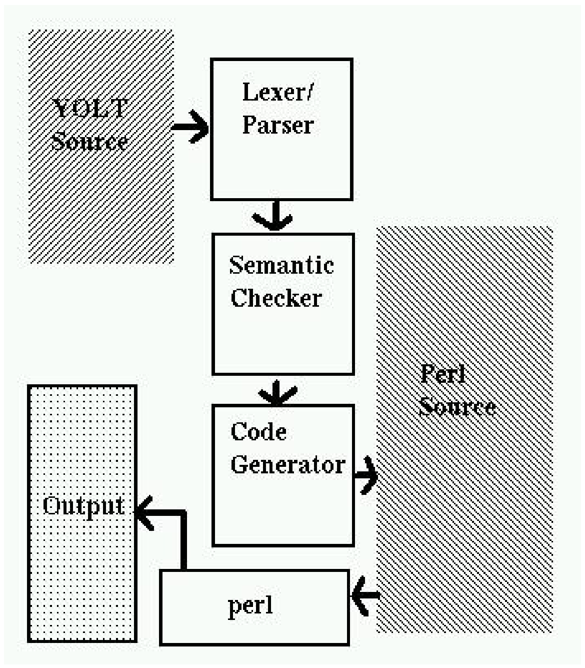


Figure 1: YOLT model of computation

7 Testing

7.1 Phase I

This testing phase involved initial phases of development, individual development, and mainly programmer testing based on an incremental implementation of functionality in each individual module. As specific methods or functionality were implemented small tests would be conducted in order to assure proper behavior from a specific feature or element.
Additionally, this is where the automation scripts and setups were done to prepare for batch scripting test cases and individual YOLT test files. In this regard, the first phase of the project involved not a great deal of systemic testing applications, rather one-on-one feedback between tester and programmer, or programmer and the group.

7.2 Phase II

This phase is essentially about testing of Alpha versions of complete modules, interfaces, etc. Here, these are already defined between the main modules of the interpreter. Testing involved inputting known to be good inputs to observe functionality of the separate modules such as the Lexer/Parser, Semantic Checker, Code Generator. Initially only the Lexer/Parser was tested throughout in this phase as the other elements were not yet ready for full interface testing. Here the automation of the Lexer/Parser cases happened, trying known to be good and known to be bad inputs on the Lexer/Parser and observing output behavior vs. reference files.

7.3 Phase III

During the third phase of testing the challenge was integration. Here the main modules and components are basically in their Beta states. putting in known good inputs to observer interaction between modules, as well as known bad modules to observe Lexer/Parser, Semantic Checker functionality and error detection.

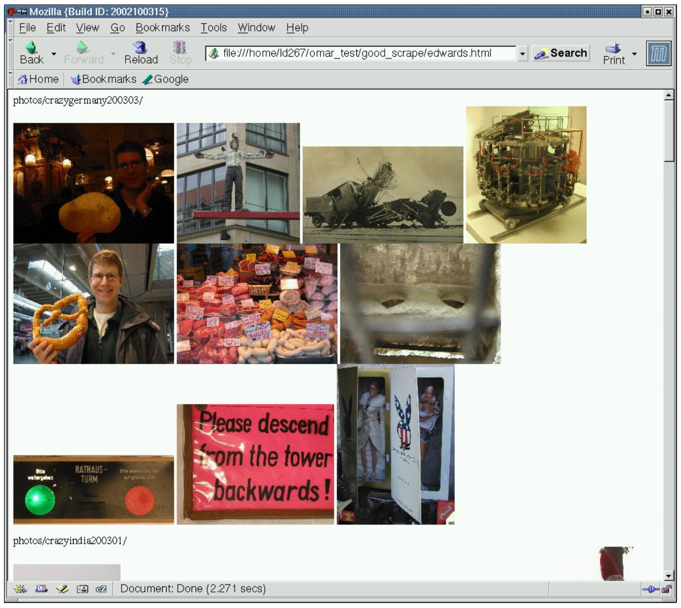

Mozilla {Build ID: 2002100315}
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///home/ld267/omar_test/good_scrape/edwards.html
Search Print
Home Bookmarks Google
photos/crazygermany200303/
Please descend from the tower backwards!
RATHAUS-TURM
photos/crazyindia200301/
Document: Done (2.271 secs)

Figure 2: Example of information scraping, Prof. Edwards' Crazy Pictures Page, generated by the YOLT edwards.y program

7.3.1 Scripting and Automation

The nature of any interpreter such as YOLT is that the input to the interpreter can be of an infinite variety. From basic echo to large information scarping programs, the testing methodology cannot hope to cover all interpreter input possibilities. Our hope was to test all desired features in YOLT in the three phases described above. Where as the input of

Lexer/Parser The input of the Lexer/Parser is the best suited to automation and batch testing. In this regard, we do not care about the end result of the code, nor of its semantic structure. We are only concerned with generating the corresponding (correct) AST in preparation for the semantic checker. This testing was automated with test cases that were known to be good.

Semantic Checker The input of the semantic checker was also scripted for batch runs. Known good and known-bad cases were run against common semantic errors with the resultant file documented.

Code Generator The Code Generator was much harder to test in the same way as Semantic Checker or Lexer/Parser. Here, input was put into the CodeGenerator (known-good input) and the resulting Perl code tested for functionality. We found that this approach worked best, as we tried to write YOLT programs to achieve a specific result. This approach allowed us to spot both programming errors, as well as design decisions that had to be evaluated such as the ”gets” behavior. Testing code generation in this way allowed us to see YOLT functionality for the very first time. As more and more YOLT features were exercised in these test programs, they were run through the whole interpreter and generated the desired output. This allowed us to test all three components as a whole, and observe the output Perl code in action.

7.3.2 Common Syntax Errors

Testing the Lexer/Parser aimed at solving common syntax errors (errors where the program will not compile) due to generation of a malformed AST. A simple batch test suit was developed to test common known good cases, as well as common know bad cases of Lexer/Parser input.

- Escape Characters
- Perl Blocks
- No begin or end statement.
- Semicolon after begin with a function declaration afterwards
- Forgotton the semicolon (;) in the end of a statement.
- A reserved word with a capital letter (If instead of if, etc).
- Text string with a quote ” but did not end with ”.
- No matching parenthesis (), braces or brackets [].

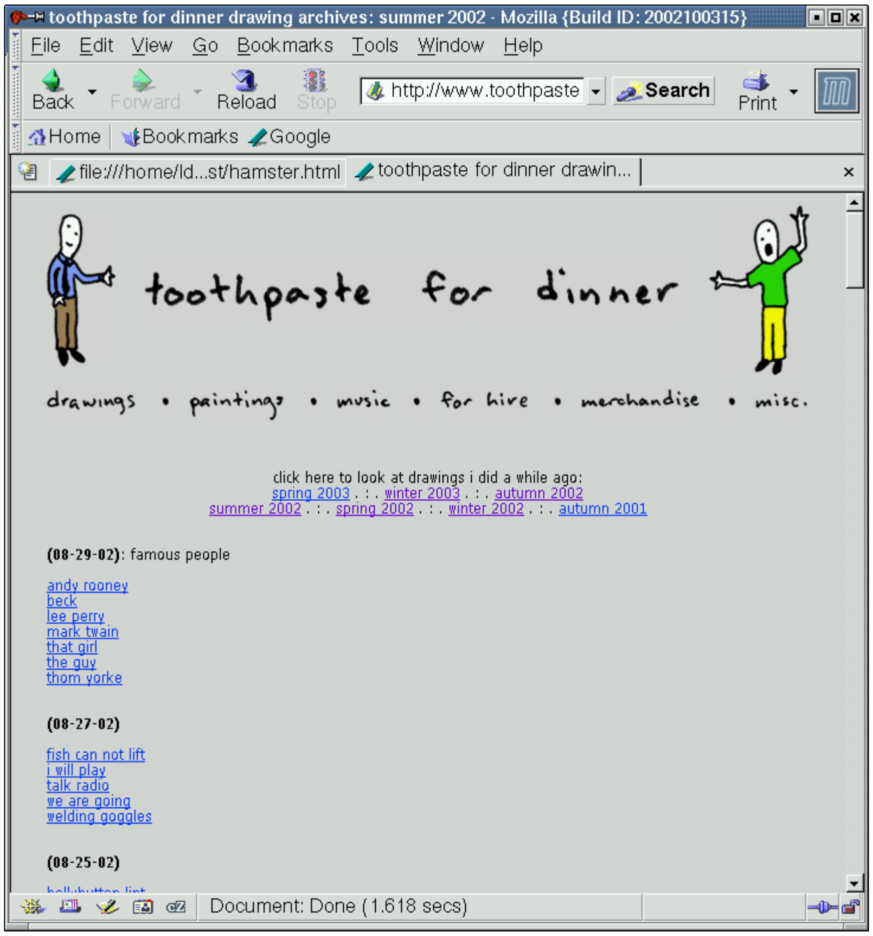
7.3.3 Common Semantic Errors

The semantic checker phase of the YOLT interpreter cycle deals with common and trivial semantic errors (where the program will not behave as expected.) In this regard, a series of test-programs was developed to test common errors in the YOLT language, as well as observe the output and behavior of the semantic checker. The most common trivial semantic errors include:

- Structural tests (malformed if, etc)
- Used = instead of == in a comparison.
- Used a variable before assigning a value to it.
- Calling function that has not been declared

7.4 Test Programs, Integration

The majority of these test cases were programs written on the fly to test the complete YOLT package. The following YOLT program is one example of such programs.
The goal here was to go to a webcomics site and extract any comics that had the word "hamster" in the URL. The target site was "http://www.toothpastefordinner.com," the Summer 2002 archive.


toothpaste for dinner drawing archives: summer 2002 - Mozilla {Build ID: 2002100315}
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.toothpaste
Search
Print
Home Bookmarks Google
file:///home/ld...st/hamster.html
toothpaste for dinner drawin...
toothpaste for dinner
drawings • paintings • music • for hire • merchandise • misc.
Document: Done (1.618 secs)

click here to look at drawings i did a while ago:
spring 2003 . : . winter 2003 . : . autumn 2002
summer 2002 . : . spring 2002 . : . winter 2002 . : . autumn 2001

(08-29-02): famous people

andy rooney
beck
lee perry
mark twain
that girl
the guy
thom yorke

(08-27-02)

fish can not lift
i will play
talk radio
we are going
welding goggles

(08-25-02)

Figure 3: Source website, www.toothpastefordinner.com

ORIGINAL YOLT PROGRAM:

```
# @author: Lukas Dudkowski
# get stuff from toothpastefordinner
# used for integration testing, shown during presentation

begin

$toothpaste_home="http://www.toothpastefordinner.com/";
$toothpaste:="http://www.toothpastefordinner.com/archives-sum02.p
hp";

$tags="<a href=\"(.*)\">.*hamster.*</a>";

$elements = $tags @ $toothpaste;

foreach $x in $elements {

  echo "<img src=\"".$toothpaste_home.$x."\""."><br>";
  echo "\n";

}

end
```

GENERATED PERL CODE:

```
$toothpaste_home ="http://www.toothpastefordinner.com/";
system('wget -q -O - http://www.toothpastefordinner.com/archives-
sum02.php  > toothpaste.txt');
open INFILE, "toothpaste.txt";
@toothpaste=<INFILE>;
close INFILE;
system ('rm toothpaste.txt');
$toothpaste = "http://www.toothpastefordinner.com/archives-sum02.
php";
$tags ="<a href=\"(.*)\">.*hamster.*</a>";
@tmp1=();
foreach ( @toothpaste) {
if ($_=~m/($tags)/i){
push @tmp1, $2}
}
@elements = @tmp1;
foreach $x ( @elements ) {
print "<img src=\"".$toothpaste_home.$x."\""."><br>";
print "\n";
}
```

RESULTANT HTML FILE:

```
<img src="http://www.toothpastefordinner.com/072802/hamster-table-tennis.gif"><br>
<img src="http://www.toothpastefordinner.com/072502/even-hamsters.gif"><br>
<img src="http://www.toothpastefordinner.com/060602/hamsters-are-the-best.gif"><br>
```

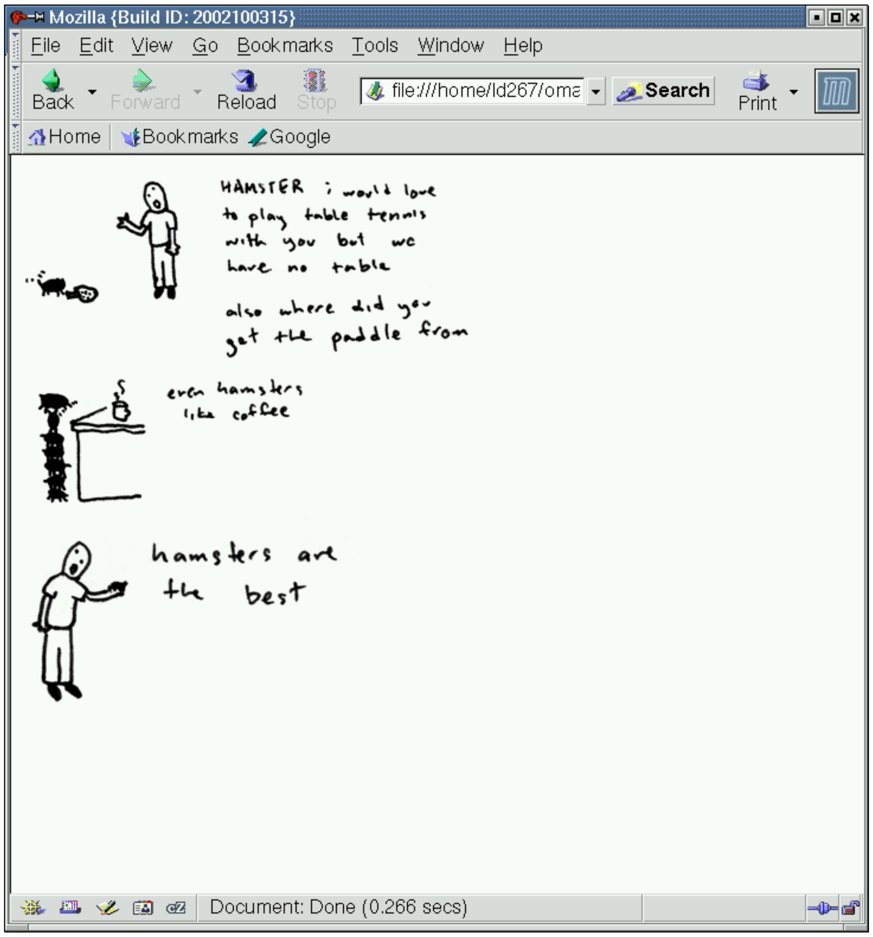

Mozilla {Build ID: 2002100315}
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///home/ld267/oma
Search
Print
Home Bookmarks Google
HAMSTER i would love to play table tennis with you but we have no table
also where did you get the paddle from
even hamsters like coffee
hamsters are the best
Document: Done (0.266 secs)

Figure 4: Resultant HTML file

7.4.1 Lexer/Parser Known Good Cases

A series of test cases was run on the Lexer/Parser and the resultant output compared to that of a hand-generated "expected" output. These were known-good cases where we did not expect to generate any problems with the Lexer/Parser. The process was automated using a Bash script, and the "reference" hand-generated file was compared to the Lexer/Parser generated file. The following lists the features tested using this method, the suffix of the file name explains the tested functionality:

```
test_addition.y
test_comments.y
test_division.y
test_eq_bad.y
test_eq.y
test_escape.y
test_function.y
test_gt_eq.y
test_gt.edwards.ytest_addition.y
test_comments.y
test_division.y
test_eq_bad.y
test_eq.y
test_escape.y
test_function.y
test_gt_eq.y
test_gt.y
test_logical_and.y
test_logical_or.y
test_lt_eq.y
test_lt.y
test_multiplication.y
test_not_eq.y
test_no_whitespace.y
test_null.y
test_perl.y
test_subtraction.y
```

***** THIS FILE IS AUTO GENERATED ******

```
############# TEST CASES BEGIN HERE ############

YOLT source file: test_addition.y

begin
$a=$a+5;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $a ( + $a 5 ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( + $a 5 ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_comments.y

begin

#comments go here

$y=10;
echo $y;

#more comments
#more comments here

echo $y;
$z=8;

#more comments

end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $y 10 ) ( echo $y ) ( echo $y ) ( = $z 8 ) )
```

```
#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $y 10 ) ( echo $y ) ( echo $y ) ( = $z 8 ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_division.y

begin
$x=$x/   65;
end

#######TREE LEXER/PARSER GENERATED###############
 ( begin ( = $x ( / $x 65 ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $x ( / $x 65 ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_eq_bad.y

begin
$a = $b == $c;

#######TREE LEXER/PARSER GENERATED###############
line 3:1: expecting "end", found 'null'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( == $b $c ) ) )


##################################################
Diff with REFERENCE:
1c1,3
<  ( begin ( = $a ( == $b $c ) ) )
---
```

```
> line 3:1: expecting "end", found 'null'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)


##################################################
YOLT source file: test_eq.y

begin
$a = $b == $c;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $a ( == $b $c ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( == $b $c ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_escape.y

begin
echo "\n\r\t\b\f\"\\\'\\\$\.\/";
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( echo \n\r\t\b\f\"\\\'\\\$\.\/ ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( echo \n\r\t\b\f\"\\\'\\\$\.\/ ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_function.y

begin
```

```
declare my_function();

function my_function(){

# do nothing

}

call my_function();

end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( declare my_function ) ( function my_function ( { } ) )
 ( call my_function ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( declare my_function ) ( function my_function ( { } ) )
 ( call my_function ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_gt_eq.y

begin
$a = $b >= $c;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $a ( >= $b $c ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( >= $b $c ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_gt.y
```

```
begin
$a = $b > $c;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $a ( > $b $c ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( > $b $c ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_logical_and.y

begin
$a = $b && $c;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $a ( && $b $c ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( && $b $c ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_logical_or.y

begin
$d = $e || $f;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $d ( || $e $f ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $d ( || $e $f ) ) )
```

```
##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_lt_eq.y

begin
$a = $b <= $c;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $a ( <= $b $c ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( <= $b $c ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_lt.y

begin
$a = $b < $c;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $a ( < $b $c ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( < $b $c ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_multiplication.y

begin
$z=$y*3;
```

```
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $z ( * $y 3 ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $z ( * $y 3 ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_not_eq.y

begin
$a = $b != $c;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $a ( != $b $c ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a ( != $b $c ) ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_null.y

begin
;
;
end

########TREE LEXER/PARSER GENERATED###############
 begin

#########  HAND GENERATED REFERENCE FILE #########
 begin

##################################################
```

```
Diff with REFERENCE:


##################################################
YOLT source file: test_perl.y

begin

#embed some perl
&{


  ~!@#$%^&*()_+}{:"><?,./;'p[]=-
  ABCDEFGHIJKLMN
@^@$(!@$&!@)$(!@)($*!@$
#########################
?>?>?>":":}{P{}P
  #comment in Perl
  print "Hello World.\n";

}&

end

########TREE LEXER/PARSER GENERATED###############
 ( begin &{


  ~!@#$%^&*()_+}{:"><?,./;'p[]=-
  ABCDEFGHIJKLMN
@^@$(!@$&!@)$(!@)($*!@$
#########################
?>?>?>":":}{P{}P
  #comment in Perl
  print "Hello World.\n";

}& )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin &{


  ~!@#$%^&*()_+}{:"><?,./;'p[]=-
  ABCDEFGHIJKLMN
@^@$(!@$&!@)$(!@)($*!@$
#########################
?>?>?>":":}{P{}P
  #comment in Perl
  print "Hello World.\n";

}& )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: test_subtraction.y

begin
$xvd = $fdsa - 5;
end

########TREE LEXER/PARSER GENERATED###############
 ( begin ( = $xvd ( - $fdsa 5 ) ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $xvd ( - $fdsa 5 ) ) )


##################################################
Diff with REFERENCE:


##################################################
DIFF RESULTS FOR THE TEST RUN:
Parsing test_addition.y
file PASSED
Parsing test_comments.y
file PASSED
Parsing test_division.y
file PASSED
Parsing test_eq_bad.y
FAILED: output mismatch
Parsing test_eq.y
file PASSED
Parsing test_escape.y
file PASSED
Parsing test_function.y
file PASSED
Parsing test_gt_eq.y
```

```
file PASSED
Parsing test_gt.y
file PASSED
Parsing test_logical_and.y
file PASSED
Parsing test_logical_or.y
file PASSED
Parsing test_lt_eq.y
file PASSED
Parsing test_lt.y
file PASSED
Parsing test_multiplication.y
file PASSED
Parsing test_not_eq.y
file PASSED
Parsing test_null.y
file PASSED
Parsing test_perl.y
file PASSED
Parsing test_subtraction.y
file PASSED

END OF TEST


###########################################
```

7.4.2 Lexer/Parser Known Bad Cases

A series of test cases was run on the Lexer/Parser and the resultant output compared to that of a hand-generated "expected" output. These were known-bad cases where we did expected to generate problems with the Lexer/Parser. The process was automated using a Bash script, and the "reference" hand-generated file was compared to the Lexer/Parser generated file. The following lists the common syntax errors tested using this method, the suffix of the file name explains the tested error type.

```
bad_test_begin_semi.y
bad_test_capital_reserve.y
bad_test_end_semi.y
bad_test_no_begin.y
bad_test_no_end.y
bad_test_no_left_endquote.y
bad_test_no_matching_left_braces.y
bad_test_no_matching_left_bracket.y
bad_test_no_matching_left_parens.y
bad_test_no_matching_right_braces.y
bad_test_no_matching_right_bracket.y
bad_test_no_matching_right_parens.y
bad_test_no_right_endquote.y
bad_test_no_semi.y
```

***** THIS FILE IS AUTO GENERATED ******

```
############# TEST CASES BEGIN HERE ############

YOLT source file: bad_test_begin_semi.y

begin;

declare func();

  echo "Hello world.";
end

########TREE LEXER/PARSER GENERATED###############
line 3:1: expecting "end", found 'declare'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( declare func ) ( echo Hello world. ) )


##################################################
Diff with REFERENCE:
1c1,3
<  ( begin ( declare func ) ( echo Hello world. ) )
---
> line 3:1: expecting "end", found 'declare'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)


##################################################
YOLT source file: bad_test_capital_reserve.y

begin
$a=1;
#using capital letters for reserved words

If ($a > 0){
  echo "hello world.";
}
end
```

```
########TREE LEXER/PARSER GENERATED###############
line 5:1: expecting "end", found 'If'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) } )
 ) )


##################################################
Diff with REFERENCE:
1c1,3
<  ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) }
 ) ) )
---
> line 5:1: expecting "end", found 'If'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)


##################################################
YOLT source file: bad_test_end_semi.y

begin
  echo "Hello world.";
end;

########TREE LEXER/PARSER GENERATED###############
line 3:4: expecting EOF, found ';'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( echo Hello world. ) )


##################################################
Diff with REFERENCE:
1c1,3
<  ( begin ( echo Hello world. ) )
---
> line 3:4: expecting EOF, found ';'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)
```

```
##################################################
YOLT source file: bad_test_no_begin.y


  echo "Hello world.";
end

########TREE LEXER/PARSER GENERATED###############
line 2:3: expecting "begin", found 'echo'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( echo Hello world. ) )


##################################################
Diff with REFERENCE:
1c1,3
<  ( begin ( echo Hello world. ) )
---
> line 2:3: expecting "begin", found 'echo'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)


##################################################
YOLT source file: bad_test_no_end.y

begin
  echo "Hello world.";


########TREE LEXER/PARSER GENERATED###############
line 4:1: expecting "end", found 'null'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( echo Hello world. ) )


##################################################
Diff with REFERENCE:
1c1,3
```

```
<  ( begin ( echo Hello world. ) )
---
> line 4:1: expecting "end", found 'null'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)


##################################################
YOLT source file: bad_test_no_left_endquote.y

begin
$a=1;
#no ending quotes in string

if ($a > 0){
  echo hello world.";
}
end

########TREE LEXER/PARSER GENERATED###############
line 6:8: unexpected token: hello
Exception in thread "main" java.lang.OutOfMemoryError

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) } )
 ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: bad_test_no_matching_left_braces.y

begin
$a=1;
#no matching braces

if ($a > 0)
  echo "hello world.";
}
end

########TREE LEXER/PARSER GENERATED###############
line 7:1: expecting "end", found '}'
```

```
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) } )
 ) )


##################################################
Diff with REFERENCE:
1c1,3
<  ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) }
 ) ) )
---
> line 7:1: expecting "end", found '}'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)


##################################################
YOLT source file: bad_test_no_matching_left_bracket.y

begin
$a[0]=1;
#no matching bracket

$a0]=2;

end

########TREE LEXER/PARSER GENERATED###############
line 5:4: unexpected token: ]
 ( begin ( = ( $a [ 0 ] ) 1 ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a [ 0 ] 1 ) ( = $a [ 0 ] 2 ) )


##################################################
Diff with REFERENCE:
1c1,2
<  ( begin ( = $a [ 0 ] 1 ) ( = $a [ 0 ] 2 ) )
---
> line 5:4: unexpected token: ]
>  ( begin ( = ( $a [ 0 ] ) 1 ) )
```

```
##################################################
YOLT source file: bad_test_no_matching_left_parens.y

begin
$a=1;
#no matching parens

if $a > 0)  {
  echo "hello world.";
}
end

#######TREE LEXER/PARSER GENERATED###############
line 5:4: expecting LPAREN, found '$a'
line 7:1: expecting "end", found '}'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) } )
 ) )


##################################################
Diff with REFERENCE:
1c1,4
<  ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) }
 ) ) )
---
> line 5:4: expecting LPAREN, found '$a'
> line 7:1: expecting "end", found '}'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)


##################################################
YOLT source file: bad_test_no_matching_right_braces.y

begin
$a=1;
#no matching braces

if ($a > 0)  {
  echo "hello world.";
```

```
end

#######TREE LEXER/PARSER GENERATED##############
line 8:1: expecting RCURLY, found 'end'
line 9:1: unexpected token: null
line 9:1: expecting "end", found 'null'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) } )
 ) )


################################################
Diff with REFERENCE:
1c1,5
<  ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) }
 ) ) )
---
> line 8:1: expecting RCURLY, found 'end'
> line 9:1: unexpected token: null
> line 9:1: expecting "end", found 'null'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)


################################################
YOLT source file: bad_test_no_matching_right_bracket.y

begin
$a[0]=1;
#no matching bracket

$a[0=2;

end

#######TREE LEXER/PARSER GENERATED##############
line 5:5: expecting RBRACK, found '='
line 5:7: unexpected token: ;
 ( begin ( = ( $a [ 0 ] ) 1 ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a [ 0 ] 1 ) ( = $a [ 0 ] 2 ) )
```

```
##################################################
Diff with REFERENCE:
1c1,3
<  ( begin ( = $a [ 0 ] 1 ) ( = $a [ 0 ] 2 ) )
---
> line 5:5: expecting RBRACK, found '='
> line 5:7: unexpected token: ;
>  ( begin ( = ( $a [ 0 ] ) 1 ) )


##################################################
YOLT source file: bad_test_no_matching_right_parens.y

begin
$a=1;
#no matching parens

if ($a > 0  {
  echo "hello world.";
}
end

########TREE LEXER/PARSER GENERATED###############
line 5:13: expecting RPAREN, found '{'
line 7:1: expecting "end", found '}'
java.lang.NullPointerException
at Main.main(Main.java:23)

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) } )
 ) )


##################################################
Diff with REFERENCE:
1c1,4
<  ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) }
 ) ) )
---
> line 5:13: expecting RPAREN, found '{'
> line 7:1: expecting "end", found '}'
> java.lang.NullPointerException
>  at Main.main(Main.java:23)
```

```
##################################################
YOLT source file: bad_test_no_right_endquote.y

begin
$a=1;
#no ending quotes in string

if ($a > 0){
  echo "hello world.;
}
end

########TREE LEXER/PARSER GENERATED###############
Exception in thread "main" java.lang.OutOfMemoryError

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a 1 ) ( if ( > $a 0 ) ( { ( echo hello world. ) } )
 ) )


##################################################
Diff with REFERENCE:


##################################################
YOLT source file: bad_test_no_semi.y

begin
$a=1;
#no semi after $b
$b=2
$c=3;
end

########TREE LEXER/PARSER GENERATED###############
line 5:1: expecting SEMI, found '$c'
 ( begin ( = $a 1 ) )

#########  HAND GENERATED REFERENCE FILE #########
 ( begin ( = $a 1 ) ( = $b 2 ) ( = $c 3 ) )


##################################################
Diff with REFERENCE:
1c1,2
<  ( begin ( = $a 1 ) ( = $b 2 ) ( = $c 3 ) )
```

```
---
> line 5:1: expecting SEMI, found '$c'
>  ( begin ( = $a 1 ) )


##################################################
DIFF RESULTS FOR THE TEST RUN:
Parsing bad_test_begin_semi.y
FAILED: output mismatch
Parsing bad_test_capital_reserve.y
FAILED: output mismatch
Parsing bad_test_end_semi.y
FAILED: output mismatch
Parsing bad_test_no_begin.y
FAILED: output mismatch
Parsing bad_test_no_end.y
FAILED: output mismatch
Parsing bad_test_no_left_endquote.y
FAILED: yolt lexer/parser terminated
Parsing bad_test_no_matching_left_braces.y
FAILED: output mismatch
Parsing bad_test_no_matching_left_bracket.y
FAILED: output mismatch
Parsing bad_test_no_matching_left_parens.y
FAILED: output mismatch
Parsing bad_test_no_matching_right_braces.y
FAILED: output mismatch
Parsing bad_test_no_matching_right_bracket.y
FAILED: output mismatch
Parsing bad_test_no_matching_right_parens.y
FAILED: output mismatch
Parsing bad_test_no_right_endquote.y
FAILED: yolt lexer/parser terminated
Parsing bad_test_no_semi.y
FAILED: output mismatch

END OF TEST


###########################################
```

7.4.3 Semantic Known-Good/Bad Cases

A series of test cases was run on the Semantic Checker and the resultant output put in the file. These were known-bad cases where we did expected to generate problems with the Semantic Checker. A "known-good" file was run as well, with that particular error corrected, and the output of the Semantic Checker documented. The process was automated using a Bash script, and the resultant file showed common semantic errors detected by the Semantic Checker, as well as proper files being "passed" on to the Code Generator. The following lists the common semantic errors tested using this method, the suffix of the file name explains the tested error type. The "fix" file is the file with the corrected error.

```
semantic_test_assign_fix.y
semantic_test_assign.y
semantic_test_malformed_fix.y
semantic_test_malformed.y
semantic_test_no_func_fix.y
semantic_test_no_func.y
semantic_test_no_init_fix.y
semantic_test_no_inits_fix.y
semantic_test_no_inits.y
semantic_test_no_init.y
```

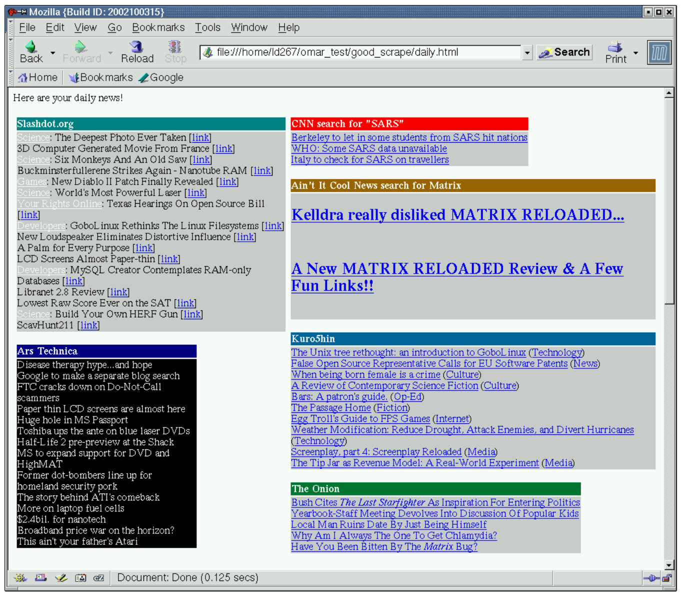
```
***** THIS FILE IS AUTO GENERATED ******

\bigskip


\begin{verbatim}

############# TEST CASES BEGIN HERE ############

YOLT source file: semantic_test_assign_fix.y

begin

$x=1;

if ($x==2){

  echo "hello world.";

}

end


##################################################
TYPE CHECKING DONE WITH NO ERRORS


##################################################
YOLT source file: semantic_test_assign.y

begin

$x=1;

if ($x=2){

  echo "hello world.";

}

end


##################################################
```

```
line 5:7: expecting RPAREN, found '='
line 9:1: expecting "end", found '}'
java.lang.NullPointerException
at Semantics.checkSemantics(Semantics.java:26)
at Main.main(Main.java:37)


##################################################
YOLT source file: semantic_test_malformed_fix.y

begin

$x=1;

if ($x==2){

  echo "hello world.";
}

end


##################################################
TYPE CHECKING DONE WITH NO ERRORS


##################################################
YOLT source file: semantic_test_malformed.y

begin

$x=1;

if ($x==2){

  echo "hello world.";
}
}

end


##################################################
line 9:1: expecting "end", found '}'
java.lang.NullPointerException
at Semantics.checkSemantics(Semantics.java:26)
at Main.main(Main.java:37)


##################################################
YOLT source file: semantic_test_no_func_fix.y

begin

declare my_func();

call my_func();

my_func(){

}

end


##################################################
line 7:1: expecting "end", found 'my_func'
java.lang.NullPointerException
at Semantics.checkSemantics(Semantics.java:26)
at Main.main(Main.java:37)


##################################################
YOLT source file: semantic_test_no_func.y

begin

call my_func();

my_func(){

}

end


##################################################
line 5:1: expecting "end", found 'my_func'
java.lang.NullPointerException
at Semantics.checkSemantics(Semantics.java:26)
at Main.main(Main.java:37)
```

```
##################################################
YOLT source file: semantic_test_no_init_fix.y

begin

$x=1;

echo $x;

end


##################################################
TYPE CHECKING DONE WITH NO ERRORS


##################################################
YOLT source file: semantic_test_no_inits_fix.y

begin

$x=1;
$y=1;

echo $x;
echo $y;

end


##################################################
TYPE CHECKING DONE WITH NO ERRORS


##################################################
YOLT source file: semantic_test_no_inits.y

begin

echo $x;
echo $y;

end
```

```
##################################################
error: the variable $x has not been defined
error: echo does not have a valid expression
ERROR IN SEMANTIC CHECKING!
error in semantic checking, exiting compilation


##################################################
YOLT source file: semantic_test_no_init.y

begin

echo $x;

end


##################################################
error: the variable $x has not been defined
error: echo does not have a valid expression
ERROR IN SEMANTIC CHECKING!
error in semantic checking, exiting compilation


##################################################
```

7.4.4 Features Testing


Mozilla {Build ID: 2002100315}
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
file:///home/ld267/omar_test/good_scrape/daily.html
Search
Print
Home Bookmarks Google

Here are your daily news!

Slashdot.org

Science: The Deepest Photo Ever Taken [link]
3D Computer Generated Movie From France [link]
Science: Six Monkeys And An Old Saw [link]
Buckminsterfullerene Strikes Again - Nanotube RAM [link]
Games: New Diablo II Patch Finally Revealed [link]
Science: World's Most Powerful Laser [link]
Your Rights Online: Texas Hearings On Open Source Bill [link]
Developers: GoboLinux Rethinks The Linux Filesystems [link]
New Loudspeaker Eliminates Distortive Influence [link]
A Palm for Every Purpose [link]
LCD Screens Almost Paper-thin [link]
Developers: MySQL Creator Contemplates RAM-only Databases [link]
Libranet 2.8 Review [link]
Lowest Raw Score Ever on the SAT [link]
Science: Build Your Own HERF Gun [link]
ScavHunt211 [link]

Ars Technica

Disease therapy hype...and hope
Google to make a separate blog search
FTC cracks down on Do-Not-Call scammers
Paper thin LCD screens are almost here
Huge hole in MS Passport
Toshiba ups the ante on blue laser DVDs
Half-Life 2 pre-preview at the Shack
MS to expand support for DVD and HighMAT
Former dot-bombers line up for homeland security pork
The story behind ATI's comeback
More on laptop fuel cells
$2.4bil. for nanotech
Broadband price war on the horizon?
This ain't your father's Atari

CNN search for "SARS"

Berkeley to let in some students from SARS hit nations
WHO: Some SARS data unavailable
Italy to check for SARS on travellers

Ain't It Cool News search for Matrix

Kelldra really disliked MATRIX RELOADED...

A New MATRIX RELOADED Review & A Few Fun Links!!

Kuro5hin

The Unix tree rethought: an introduction to GoboLinux (Technology)
False Open Source Representative Calls for EU Software Patents (News)
When being born female is a crime (Culture)
A Review of Contemporary Science Fiction (Culture)
Bars: A patron's guide. (Op-Ed)
The Passage Home (Fiction)
Egg Troll's Guide to FPS Games (Internet)
Weather Modification: Reduce Drought, Attack Enemies, and Divert Hurricanes (Technology)
Screenplay, part 4: Screenplay Reloaded (Media)
The Tip Jar as Revenue Model: A Real-World Experiment (Media)

The Onion

Bush Cites *The Last Starfighter* As Inspiration For Entering Politics
Yearbook-Staff Meeting Devolves Into Discussion Of Popular Kids
Local Man Ruins Date By Just Being Himself
Why Am I Always The One To Get Chlamydia?
Have You Been Bitten By The *Matrix* Bug?

Document: Done (0.125 secs)

Figure 5: Example of information scraping, Daily News site compiled from various news sources.

8 Source and Target Language Examples

8.1 daily.y to daily.pl

```
# Principal Author: T. Mark Kuba
#
# scrape daily news and headlines
#

begin

echo "Here are your daily news!<p>";

echo "<p>";

#big table
echo "<TABLE BORDER=0><TR><TD VALIGN=\"TOP\">";

##################
# slashdot
```

```
$slashdot := "http://slashdot.org";

$tags = "<a href=\"(.*)\" TITLE=\"\"><b>Read More...</b></a>";

$story_headline = "FACE=\"arial,helvetica\" SIZE=\"4\" COLOR=\"#F
FFFFF\"><b>(.*)</b></font></td>";

$array = $tags @ $slashdot;
$array2 = $story_headline @ $slashdot;

$x = 0;

echo "<table border=0>";
echo "<tr><td bgcolor=\"#008080\"><font color=\"#FFFFFF\"><strong
>Slashdot.org</strong></font></td></tr>";
echo "<tr><td bgcolor=\"#CCCCCC\">";

foreach $element in $array {

  echo "  " .$array2[$x];
  echo "  [<a href=\"". "http:" . $element . "\">link</a>]<br>";
  $x = $x+1;
}

echo "</td></tr></table>";


echo "<p>";

##################
# ars technica


$ars := "http://www.arstechnica.com";

$tags = "<h2>(.*)</h2>";

$array = $tags @ $ars;

echo "<table border=0><tr><td bgcolor=\"#000080\"><font color=\"#
FFFFFF\"><strong>Ars Technica</strong></font></td></tr>";

echo "<tr><td bgcolor=\"#000000\">";
foreach $element in $array {
  echo $element . "<br>";

}
echo "</td></tr></table>";

echo "</TD><TD VALIGN=\"TOP\">";

##################
# cnn search

$cnn := "http://www.cnn.com";

$search_term = "SARS";

$story_tags = "<a href=\"(/2003/.*)\">.*" . $search_term. ".*</a>
";

$story_tags2 = "<a href=\"/2003/.*\">(.*" .$search_term .".*)</a>
";

$myArray = $story_tags @ $cnn;
$myArray2 = $story_tags2 @ $cnn;


$x = 0;

echo "<table border=0><tr><td bgcolor=\"#FF0000\"><font color=\"#
FFFFFF\"><strong>CNN search for \"" . $search_term . "\"</strong>
</font></td></tr>";

echo "<tr><td bgcolor=\"#CCCCCC\">";
foreach $el in $myArray {

  echo "<a href=\"" . $cnn . $el . "\">" . $myArray2[$x]  ."</a><
br>";
  $x = $x+1;

}
echo "</td></tr></table>";

echo "<p>";

#################
# aint it cool news
```

```
$aicn := "http://www.aintitcoolnews.com/";

$search_term = "Matrix";

$tags = "<a href=\"(display\.cgi.*)\">.*" . $search_term . ".*</a
>";
$tags2 = "<a href=\"display\.cgi.*\">(.*" . $search_term . ".*)</
a>";

$a1 = $tags @ $aicn;
$a2 = $tags2 @ $aicn;


echo "<table border=0><tr><td bgcolor=\"#996600\"><font color=\"#
FFFFFF\"><strong>Ain't It Cool News search for " . $search_term .
"</strong></font></td></tr>";

echo "<tr><td bgcolor=\"#CCCCCC\">";

$x = 0;
foreach $el in $a1 {

  echo "<a href=\"" . $aicn . $el . "\">" . $a2[$x] . "</a><br>";
  $x = $x + 1;

}
echo "</td></tr></table>";

echo "<p>";

################
# kuro5hin


$k5 := "http://www.kuro5hin.org";

$tags = "<font FACE=\"verdana, arial, helvetica, sans-serif\"><b>
<a HREF=\"(/story/.*)\" style=\"text-decoration:none;\"><font col
or=\"#000000\">.*</font>";

$tags2 = "<font FACE=\"verdana, arial, helvetica, sans-serif\"><b
><a HREF=\"/story/.*\" style=\"text-decoration:none;\"><font colo
r=\"#000000\">(.*)</font>";


$ar1 = $tags @ $k5;
```

```
$ar2 = $tags2 @ $k5;


echo "<table border=0><tr><Td bgcolor=\"#006699\"><font color=\"#
FFFFFF\"><strong>Kuro5hin</strong></font></td></tr>";
echo "<tr><td bgcolor=\"#CCCCCC\">";
$x = 0;
foreach $el in $ar1 {

  echo "<a href=\"" . $k5 . $el . "\">" . $ar2[$x] . "</a><br>\n"
;
  $x = $x+1;
}
echo "</td></tr></table>";

echo "<p>";

################
# The Onion


$onion := "http://www.theonion.com";

$tags1 = "<a href=\"(/onion3917/.*)\" CLASS=\"headline\"><b>.*</b
>";

$tags2 = "<a href=\"/onion3917/.*\" CLASS=\"headline\"><b>(.*)</b
>";

$a1 = $tags1 @ $onion;
$a2 = $tags2 @ $onion;

$x = 0;

echo "<table border=0><tr><Td bgcolor=\"#007733\"><font color=\"#
FFFFFF\"><strong>The Onion</strong></font></td></tr>";
echo "<tr><td bgcolor=\"#CCCCCC\">";
foreach $el in $a1 {
  echo "<a href=\"" . $onion . $el . "\">" . $a2[$x] ."</a><br>\n
";
  $x = $x + 1;
}
echo "</td></tr></table>";

echo "</TD></TR></TABLE>";
echo "<p>";



################
# web comics
$goats := "http://www.goats.com";
$tags = "<img src=\"(/comix.*)\">";

$array = $tags @ $goats;

echo "<h5><a href=\"" .$goats . "\">goats</a></h5>";
foreach $element in $array {
  echo "<img src=\"" . $goats . $element . "\">";
}

echo "<p>";

$dilbert := "http://www.dilbert.com";
$tags = "<img src=\"(/comics/dilbert/archive/images/dilbert200.*)
\">";

$array = $tags @ $dilbert;

echo "<h5><a href=\"" .$dilbert . "\">Dilbert</a></h5>";
foreach $element in $array {
  echo "<img src=\"" . $dilbert . $element . "\">";
}




end
```

```
print "Here are your daily news!<p>";
print "<p>";
print "<TABLE BORDER=0><TR><TD VALIGN=\"TOP\">";
system('wget -q -O - http://slashdot.org  > slashdot.txt');
open INFILE, "slashdot.txt";
@slashdot=<INFILE>;
close INFILE;
system ('rm slashdot.txt');
$slashdot = "http://slashdot.org";
$tags ="<a href=\"(.*)\" TITLE=\"\"><b>Read More...</b></a>";
$story_headline ="FACE=\"arial,helvetica\" SIZE=\"4\" COLOR=\"#FF
FFFF\"><b>(.*)</b></font></td>";
@tmp1=();
foreach ( @slashdot) {
if ($_=~m/($tags)/i){
push @tmp1, $2}
}
@array = @tmp1;
@tmp2=();
foreach ( @slashdot) {
if ($_=~m/($story_headline)/i){
push @tmp2, $2}
}
@array2 = @tmp2;
$x = 0;
print "<table border=0>";
print "<tr><td bgcolor=\"#008080\"><font color=\"#FFFFFF\"><stron
g>Slashdot.org</strong></font></td></tr>";
print "<tr><td bgcolor=\"#CCCCCC\">";
foreach $element ( @array ) {
print "  ".@array2[$x];
print "  [<a href=\""."http:".$element."\">link</a>]<br>";
$x = ($x+1);
}
print "</td></tr></table>";
print "<p>";
system('wget -q -O - http://www.arstechnica.com  > ars.txt');
open INFILE, "ars.txt";
@ars=<INFILE>;
close INFILE;
system ('rm ars.txt');
$ars = "http://www.arstechnica.com";
$tags ="<h2>(.*)</h2>";
@tmp3=();
foreach ( @ars) {
if ($_=~m/($tags)/i){
```

```
push @tmp3, $2}
}
@array = @tmp3;
print "<table border=0><tr><td bgcolor=\"#000080\"><font color=\"
#FFFFFF\"><strong>Ars Technica</strong></font></td></tr>";
print "<tr><td bgcolor=\"#000000\">";
foreach $element ( @array ) {
print $element."<br>";
}
print "</td></tr></table>";
print "</TD><TD VALIGN=\"TOP\">";
system('wget -q -O - http://www.cnn.com  > cnn.txt');
open INFILE, "cnn.txt";
@cnn=<INFILE>;
close INFILE;
system ('rm cnn.txt');
$cnn = "http://www.cnn.com";
$search_term ="SARS";
$story_tags = "<a href=\"(/2003/.*)\">.*".$search_term.".*</a>";
$story_tags2 = "<a href=\"/2003/.*\">(.*".$search_term.".*)</a>";
@tmp4=();
foreach ( @cnn) {
if ($_=~m/($story_tags)/i){
push @tmp4, $2}
}
@myArray = @tmp4;
@tmp5=();
foreach ( @cnn) {
if ($_=~m/($story_tags2)/i){
push @tmp5, $2}
}
@myArray2 = @tmp5;
$x = 0;
print "<table border=0><tr><td bgcolor=\"#FF0000\"><font color=\"
#FFFFFF\"><strong>CNN search for \"".$search_term."\"</strong></f
ont></td></tr>";
print "<tr><td bgcolor=\"#CCCCCC\">";
foreach $el ( @myArray ) {
print "<a href=\"".$cnn.$el."\">".@myArray2[$x]."</a><br>";
$x = ($x+1);
}
print "</td></tr></table>";
print "<p>";
system('wget -q -O - http://www.aintitcoolnews.com/  > aicn.txt')
;
open INFILE, "aicn.txt";
```

```
@aicn=<INFILE>;
close INFILE;
system ('rm aicn.txt');
$aicn = "http://www.aintitcoolnews.com/";
$search_term ="Matrix";
$tags = "<a href=\"(display\.cgi.*)\">.*".$search_term.".*</a>";
$tags2 = "<a href=\"display\.cgi.*\">(.*".$search_term.".*)</a>";
@tmp6=();
foreach ( @aicn) {
if ($_=~m/($tags)/i){
push @tmp6, $2}
}
@a1 = @tmp6;
@tmp7=();
foreach ( @aicn) {
if ($_=~m/($tags2)/i){
push @tmp7, $2}
}
@a2 = @tmp7;
print "<table border=0><tr><td bgcolor=\"#996600\"><font color=\"
#FFFFFF\"><strong>Ain't It Cool News search for ".$search_term."<
/strong></font></td></tr>";
print "<tr><td bgcolor=\"#CCCCCC\">";
$x = 0;
foreach $el ( @a1 ) {
print "<a href=\"".$aicn.$el."\">".@a2[$x]."</a><br>";
$x = ($x+1);
}
print "</td></tr></table>";
print "<p>";
system('wget -q -O - http://www.kuro5hin.org  > k5.txt');
open INFILE, "k5.txt";
@k5=<INFILE>;
close INFILE;
system ('rm k5.txt');
$k5 = "http://www.kuro5hin.org";
$tags ="<font FACE=\"verdana, arial, helvetica, sans-serif\"><b><
a HREF=\"(/story/.*)\" style=\"text-decoration:none;\"><font colo
r=\"#000000\">.*</font>";
$tags2 ="<font FACE=\"verdana, arial, helvetica, sans-serif\"><b>
<a HREF=\"/story/.*\" style=\"text-decoration:none;\"><font color
=\"#000000\">(.*)</font>";
@tmp8=();
foreach ( @k5) {
if ($_=~m/($tags)/i){
push @tmp8, $2}
```

```
}
@ar1 = @tmp8;
@tmp9=();
foreach ( @k5) {
if ($_=~m/($tags2)/i){
push @tmp9, $2}
}
@ar2 = @tmp9;
print "<table border=0><tr><Td bgcolor=\"#006699\"><font color=\"
#FFFFFF\"><strong>Kuro5hin</strong></font></td></tr>";
print "<tr><td bgcolor=\"#CCCCCC\">";
$x = 0;
foreach $el ( @ar1 ) {
print "<a href=\"".$k5.$el."\">".@ar2[$x]."</a><br>\n";
$x = ($x+1);
}
print "</td></tr></table>";
print "<p>";
system('wget -q -O - http://www.theonion.com  > onion.txt');
open INFILE, "onion.txt";
@onion=<INFILE>;
close INFILE;
system ('rm onion.txt');
$onion = "http://www.theonion.com";
$tags1 ="<a href=\"(/onion3917/.*)\" CLASS=\"headline\"><b>.*</b>
";
$tags2 ="<a href=\"/onion3917/.*\" CLASS=\"headline\"><b>(.*)</b>
";
@tmp10=();
foreach ( @onion) {
if ($_=~m/($tags1)/i){
push @tmp10, $2}
}
@a1 = @tmp10;
@tmp11=();
foreach ( @onion) {
if ($_=~m/($tags2)/i){
push @tmp11, $2}
}
@a2 = @tmp11;
$x = 0;
print "<table border=0><tr><Td bgcolor=\"#007733\"><font color=\"
#FFFFFF\"><strong>The Onion</strong></font></td></tr>";
print "<tr><td bgcolor=\"#CCCCCC\">";
foreach $el ( @a1 ) {
print "<a href=\"".$onion.$el."\">".@a2[$x]."</a><br>\n";
```

```
$x = ($x+1);
}
print "</td></tr></table>";
print "</TD></TR></TABLE>";
print "<p>";
system('wget -q -O - http://www.goats.com  > goats.txt');
open INFILE, "goats.txt";
@goats=<INFILE>;
close INFILE;
system ('rm goats.txt');
$goats = "http://www.goats.com";
$tags ="<img src=\"(/comix.*)\">";
@tmp12=();
foreach ( @goats) {
if ($_=~m/($tags)/i){
push @tmp12, $2}
}
@array = @tmp12;
print "<h5><a href=\"".$goats."\">goats</a></h5>";
foreach $element ( @array ) {
print "<img src=\"".$goats.$element."\">";
}
print "<p>";
system('wget -q -O - http://www.dilbert.com  > dilbert.txt');
open INFILE, "dilbert.txt";
@dilbert=<INFILE>;
close INFILE;
system ('rm dilbert.txt');
$dilbert = "http://www.dilbert.com";
$tags ="<img src=\"(/comics/dilbert/archive/images/dilbert200.*)\
">";
@tmp13=();
foreach ( @dilbert) {
if ($_=~m/($tags)/i){
push @tmp13, $2}
}
@array = @tmp13;
print "<h5><a href=\"".$dilbert."\">Dilbert</a></h5>";
foreach $element ( @array ) {
print "<img src=\"".$dilbert.$element."\">";
}
```

8.2 edwards.y to edwards.pl

```
# Principal Author: T. Mark Kuba
#
```

```
# download "crazy" photos from Stephen Edwards's web site
#

begin

$home = "http://www1.cs.columbia.edu/~sedwards/";
$page := "http://www1.cs.columbia.edu/~sedwards/personal.html";

$tags = "<a href=\"(.*crazy.*)index\.html\">";

$a1 = $tags @ $page;

foreach $link in $a1 {

  echo "<p>" .$link . "<br>\n\n\n";

  $picpage := $home . $link;

  $tags2 = "<img src=\"(.*)\" alt=\".*\" width=\".*\" height=\".*
\">";

  $pics = $tags2 @ $picpage;

  foreach $pic in $pics {
    echo "<img src=\"" . $home.$link . $pic . "\">\n";
  }

}

end
```

```
$home ="http://www1.cs.columbia.edu/~sedwards/";
system('wget -q -O - http://www1.cs.columbia.edu/~sedwards/person
al.html  > page.txt');
open INFILE, "page.txt";
@page=<INFILE>;
close INFILE;
system ('rm page.txt');
$page = "http://www1.cs.columbia.edu/~sedwards/personal.html";
$tags ="<a href=\"(.*crazy.*)index\.html\">";
@tmp1=();
foreach ( @page) {
if ($_=~m/($tags)/i){
push @tmp1, $2}
}
@a1 = @tmp1;
foreach $link ( @a1 ) {
print "<p>".$link."<br>\n\n\n";
open(fileOUT, ">tmp.txt") or dienice ("Can't open tmp.txt for wri
ting");
print fileOUT $home.$link ;

close(fileOUT);
system('wget -q -O picpage.txt -i tmp.txt');
system('rm tmp.txt');
open INFILE, "picpage.txt";
@picpage=<INFILE>;
close INFILE;
system ('rm picpage.txt');
$picpage = "$home.$link";
$tags2 ="<img src=\"(.*)\" alt=\".*\" width=\".*\" height=\".*\">
";
@tmp2=();
foreach ( @picpage) {
if ($_=~m/($tags2)/i){
push @tmp2, $2}
}
@pics = @tmp2;
foreach $pic ( @pics ) {
print "<img src=\"".$home.$link.$pic."\">\n";
}
}
```

9 Lessons Learned

Omar Ahmed

—–think ahead: you can't have types for some things and not for others.....either there are types in a language or there are not; there is no middle ground

—–For testing to be done early, everyone needs to start at the same time . . . in our case, the Lexer/Parser had been done before the Semantics and Code Generation had even started, and we couldn't test until Semantics and Code Generation had progress

—–Communication is key: everyone NEEDS to know exactly how the language is going to function (people's assumptions can affect how they approach their part of the problem)

—–Sometimes, using ANTLR just isn't a good idea, and regular Java code can be more efficient (and easier to maintain)

—–CVS is good, but it is annoying to set up, and constant emails work well too

—–Bugs will always come up in integration

Lukas Dudkowski

The development of YOLT was a very interesting experience. Having skipped Software Engineering, it was nice being involved in a large group project as my last great hurrah of Computer Science. Being responsible for testing, I learned the importance of incremental development as well as constant saving and backing up. This comes especially handy when you type "rm *.java" at 2am.

Writing a 230+ page document in LaTeX can sometimes be worse than programming in Java, YOLT, and Perl combined.

Trying out features of other people's code can be fun and frustrating at the same time. Being responsible for integration as well as testing, I got the bird's eye view of the project on the whole and where we were with YOLT development. Sometimes this view can be rewarding, sometimes it can be depressing. No matter how much planning has gone into the design of various parts, when it comes to the integration of the whole project, bugs will crawl out of the woodwork. As Scotty said "... the more they overtake the plumbing, the easier it is to stop up the drain." [1]

T. Mark Kuba

The biggest lesson I learned was the importance of incremental development. Building the grammar slowly from the ground up, and testing every step of the way proved to be a fairly reliable and fool-proof way of development. Instead of attempting to fully debug a grammar of this size after completely building it, which would probably be impossible, I isolated each component of the grammar and tested each component separately before integrating them. Another lesson I learned was the importance of finalizing specifications early in order to integrate components later on easier. Or, failing that, keeping in constant communication with teammates as to what changes have been made in the grammar, the structure of the abstract syntax tree, etc.

Yuan Zheng After the project, I fully understand the process of developing a language. Even though our language is a very small and simple one, it requires early design, lexing and parsing, semantics checking, code generating, and testing. Working in the YOLT team, I also learned the importance of team-oriented development. A four person group is a fairly big group. In order to maximize the power of the team and reduce the inter-person communication and dependencies, we worked together in lab most of the time. This was done so the whole team could discuss major issues whenever they appeared and solve them with the best solution. We carefully designed several components of our project so the whole group could work in parallel. The interface is specified in early designing, and thus the dependencies are minimized.

10 Citations

1 **Search For Spock, The.** Dir. Nimoy, Leonard. Writ. Bennett, Harve. Paramount Studios, 1984.

11 Appendix: Complete Code Listing

11.1 Lexer/Parser

11.1.1 ANTLR Grammar

```
// Primary Author: T. Mark Kuba
//
//

class P extends Parser;
options {
k = 3;
buildAST = true;
exportVocab=P;
}

program
: "begin"^ (fn_decs)* (statement)* "end"! EOF!
;

fn_decs
    :   "declare"^ FN_NAME LPAREN! (id (COMMA! id)*)? RPAREN! SEM
I!
    ;

primitive
    :   (id)
    |   (StrConstant)
    |   (IntConstant)
    |   (LPAREN! expression RPAREN!)
    |   "call"^ FN_NAME LPAREN! (expression (COMMA! expression)*)
? RPAREN!
    ;

id
    :   (Identifier^) (LBRACK (expression) RBRACK)*
    ;

unaryExpression
    :   ((LNOT^)* primitive)
    ;

multExpr
    :   unaryExpression ((STAR^ | DIV^ | MOD^) unaryExpression)*
    ;
```

```
addExpr
    :   multExpr ((PLUS^ | MINUS^) multExpr)*
    ;

relExpr
    :   addExpr ((EQUAL^ | NOT_EQUAL^ | GT^ | GE^ | LT^ | LE^) ad
dExpr)*
    ;

boolExpr
    :   relExpr ((LOR^ | LAND^) relExpr)*
    ;

concatExpr
    :   boolExpr ((DOT^) boolExpr)*
    ;


expression
    :   concatExpr ((REGEXP^) concatExpr)*
//    |   "call"^ FN_NAME LPAREN! (expression (COMMA! expression)
*)? RPAREN!
    |   LCURLY^ (expression (COMMA! expression)*)? RCURLY!
    ;




statement
: assignstmnt
| foreachstmnt
| whilestmnt
| ifstmnt
| echostmnt
| functionstmnt
| includestmnt
| returnstmnt
| Perlstmnt
//    |   httpgetstmnt
    |   nullstmnt
    |   callstmnt
;

callstmnt
    :   "call"^ FN_NAME LPAREN! (expression (COMMA! expression)*)
```

```
? RPAREN! SEMI!
    ;


nullstmnt
    :   SEMI!
    ;

statementList
: LCURLY^ (statement)* RCURLY
| statement
;

fn_stmntlist
    :   LCURLY^ (statement)* RCURLY
    ;

assignstmnt
    :   id (ASSIGN^|HTTPGET^) expression SEMI!
    ;

//httpgetstmnt
//    :   id (HTTPGET^) expression SEMI!
//    ;

foreachstmnt
: "foreach"^ Identifier "in"! Identifier statementL
ist
;

whilestmnt
: "while"^ LPAREN! expression RPAREN! statementList
;

ifstmnt
: "if"^ LPAREN! expression RPAREN! statementList (o
ptions {warnWhenFollowAmbig=false;}: "else"! statementList)?
;

echostmnt
: "echo"^ expression SEMI!
;

functionstmnt
: "function"^ FN_NAME LPAREN! (id (COMMA! id)*)? RP
AREN! fn_stmntlist
```

```
;

includestmnt
: "include"^ LPAREN! expression RPAREN! SEMI!
;

returnstmnt
: "return"^ (expression)? SEMI!
;

//perlstmnt
// : LPERL^ (options {greedy=true;}: (.)*) RPERL
// : LPERL^ (options {greedy=true;}: (~(RPERL))*) RPER
L
//    :   LPERL^
//       (
//            options {greedy=false;}:
//            (
//                ('\r' '\n') => '\r' '\n'    {newline();}
//            |   '\r'    {newline();}
//            |   '\n'    {newline();}
//            |   ~('\r'|'\n')
//            )
//        )*
//        RPERL
//       ;

//paramlist
//    :   id (COMMA! paramlist)
//    |   id
//    ;


class L extends Lexer;

options {
testLiterals=false;    // don't automatically test for li
terals
k=4;                   // four characters of lookahead
charVocabulary='\u0003'..'\uFFFF';
exportVocab=P;
}

// OPERATORS
LPAREN : '(';
RPAREN : ')';
```

```
LBRACK : '[';
RBRACK : ']';
LCURLY : '{';
RCURLY : '}';
COMMA : ',';
DOT     : '.';
ASSIGN : '=';
HTTPGET          :   ":="    ;
EQUAL : "==" ;
LNOT : '!';
NOT_EQUAL : "!=" ;
DIV : '/';
PLUS : '+';
MINUS : '-';
STAR : '*';
MOD : '%';
GE : ">=" ;
GT : ">" ;
LE : "<=" ;
LT : '<';
LOR : "||" ;
LAND : "&&" ;
REGEXP           :   '@'     ;
SEMI : ';';
LPERL : "&{" ;
RPERL : "}&" ;

//TILDE            :   "~"     ;
//QMARK            :   "?"     ;
//BSLASH           :   "\\"     ;



Perlstmnt
// : LPERL^ (options {greedy=true;}: (.)*) RPERL
// : LPERL^ (options {greedy=true;}: (~(RPERL))*) RPER
L
    :   LPERL
        (
            options {greedy=false;}:
            (
                ('\r' '\n') => '\r' '\n'    {newline();}
            |   '\r'    {newline();}
            |   '\n'    {newline();}
            |   ~('\r'|'\n')
            )
        )*
        RPERL
        ;



IntConstant:
        (
            (Digit)+
        );

protected
ESC
: '\\'('n'|'r'|'t'|'b'|'f'|'"'|'\''|'\\'|'$'|'.'|'/
'|'('|')'|'?')
    ;

StrConstant
: '"'! (ESC|~('"'|'\\'))* '"'!
    ;


Comment:
        (
            ('#' (~('\r'|'\n'))* LineTerminator)
        )
{ $setType(Token.SKIP); }
;


// a dummy rule to force vocabulary to be all characters (except
special
//   ones that ANTLR uses internally (0 to 2)
protected
VOCAB
: '\3'..'\377'
;

FN_NAME
    options {testLiterals=true;}
    :   (Letter) (Letter|Digit|Underscore)*
    ;


Identifier:
        (
//            ('$') (Letter|Digit|Underscore)+ ('[' (Digit)* ']')
*
            ('$') (Letter|Digit|Underscore)+
        );


protected
Letter:
        (
            ('a'..'z')
        |
            ('A'..'Z')
        );

protected
Digit:
        (
            ('0'..'9')
        );

protected
Underscore:
        (
            ('_')
        );


protected
LineTerminator
    :   '\r' '\n' { newline(); }   // DOS
    |   '\n'      { newline(); }  // UNIX
    ;

WS : ( ' ' | '\t' | LineTerminator)+
        { $setType(Token.SKIP); }
    ;
```

11.1.2 P.java

```
// $ANTLR 2.7.2: "20030511-debugging.g" -> "P.java"$

import antlr.TokenBuffer;
import antlr.TokenStreamException;
import antlr.TokenStreamIOException;
import antlr.ANTLRException;
import antlr.LLkParser;
import antlr.Token;
import antlr.TokenStream;
import antlr.RecognitionException;
import antlr.NoViableAltException;
import antlr.MismatchedTokenException;
import antlr.SemanticException;
import antlr.ParserSharedInputState;
import antlr.collections.impl.BitSet;
import antlr.collections.AST;
import java.util.Hashtable;
import antlr.ASTFactory;
import antlr.ASTPair;
import antlr.collections.impl.ASTArray;

public class P extends antlr.LLkParser       implements PTokenTyp
es
 {

protected P(TokenBuffer tokenBuf, int k) {
  super(tokenBuf,k);
  tokenNames = _tokenNames;
  buildTokenTypeASTClassMap();
  astFactory = new ASTFactory(getTokenTypeToASTClassMap());
}

public P(TokenBuffer tokenBuf) {
  this(tokenBuf,3);
}

protected P(TokenStream lexer, int k) {
  super(lexer,k);
  tokenNames = _tokenNames;
  buildTokenTypeASTClassMap();
  astFactory = new ASTFactory(getTokenTypeToASTClassMap());
}

public P(TokenStream lexer) {
  this(lexer,3);
}

public P(ParserSharedInputState state) {
  super(state,3);
  tokenNames = _tokenNames;
  buildTokenTypeASTClassMap();
  astFactory = new ASTFactory(getTokenTypeToASTClassMap());
}

public final void program() throws RecognitionException,
TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST program_AST = null;

try {      // for error handling
AST tmp1_AST = null;
tmp1_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp1_A
ST);
match(LITERAL_begin);
{
_loop3:
do {
if ((LA(1)==LITERAL_declare)) {
fn_decs();
astFactory.addASTChild(cu
rrentAST, returnAST);
}
else {
break _loop3;
}

} while (true);
}
{
_loop5:
do {
if ((_tokenSet_0.member(LA(1))))
{
statement();
astFactory.addASTChild(cu
rrentAST, returnAST);
}
```

```
else {
break _loop5;
}

} while (true);
}
match(LITERAL_end);
match(Token.EOF_TYPE);
program_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_1);
}
returnAST = program_AST;
}

public final void fn_decs() throws RecognitionException,
TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST fn_decs_AST = null;

try {      // for error handling
AST tmp4_AST = null;
tmp4_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp4_A
ST);
match(LITERAL_declare);
AST tmp5_AST = null;
tmp5_AST = astFactory.create(LT(1));
astFactory.addASTChild(currentAST, tmp5_A
ST);
match(FN_NAME);
match(LPAREN);
{
switch ( LA(1)) {
case Identifier:
{
id();
astFactory.addASTChild(currentAST
, returnAST);
{
_loop9:
```

```
do {
if ((LA(1)==COMMA)) {
match(COMMA);
id();
astFactory.addAST
Child(currentAST, returnAST);
}
else {
break _loop9;
}

} while (true);
}
break;
}
case RPAREN:
{
break;
}
default:
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
match(RPAREN);
match(SEMI);
fn_decs_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_2);
}
returnAST = fn_decs_AST;
}

public final void statement() throws RecognitionException
, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST statement_AST = null;

try {      // for error handling
```

```
switch ( LA(1)) {
case Identifier:
{
assignstmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
case LITERAL_foreach:
{
foreachstmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
case LITERAL_while:
{
whilestmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
case LITERAL_if:
{
ifstmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
case LITERAL_echo:
{
echostmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
```

```
case LITERAL_function:
{
functionstmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
case LITERAL_include:
{
includestmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
case LITERAL_return:
{
returnstmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
case Perlstmnt:
{
AST tmp10_AST = null;
tmp10_AST = astFactory.create(LT(
1));
astFactory.addASTChild(currentAST
, tmp10_AST);
match(Perlstmnt);
statement_AST = (AST)currentAST.r
oot;
break;
}
case SEMI:
{
nullstmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
```

```
break;
}
case LITERAL_call:
{
callstmnt();
astFactory.addASTChild(currentAST
, returnAST);
statement_AST = (AST)currentAST.r
oot;
break;
}
default:
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = statement_AST;
}

public final void id() throws RecognitionException, Token
StreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST id_AST = null;

try {      // for error handling
{
AST tmp11_AST = null;
tmp11_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp11_
AST);
match(Identifier);
}
{
_loop22:
do {
if ((LA(1)==LBRACK)) {
AST tmp12_AST = null;
```

```
tmp12_AST = astFactory.cr
eate(LT(1));
astFactory.addASTChild(cu
rrentAST, tmp12_AST);
match(LBRACK);
{
expression();
astFactory.addASTChild(cu
rrentAST, returnAST);
}
AST tmp13_AST = null;
tmp13_AST = astFactory.cr
eate(LT(1));
astFactory.addASTChild(cu
rrentAST, tmp13_AST);
match(RBRACK);
}
else {
break _loop22;
}

} while (true);
}
id_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_4);
}
returnAST = id_AST;
}

public final void primitive() throws RecognitionException
, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST primitive_AST = null;

try {      // for error handling
switch ( LA(1)) {
case Identifier:
{
{
id();
```

```
astFactory.addASTChild(currentAST
, returnAST);
}
primitive_AST = (AST)currentAST.r
oot;
break;
}
case StrConstant:
{
{
AST tmp14_AST = null;
tmp14_AST = astFactory.create(LT(
1));
astFactory.addASTChild(currentAST
, tmp14_AST);
match(StrConstant);
}
primitive_AST = (AST)currentAST.r
oot;
break;
}
case IntConstant:
{
{
AST tmp15_AST = null;
tmp15_AST = astFactory.create(LT(
1));
astFactory.addASTChild(currentAST
, tmp15_AST);
match(IntConstant);
}
primitive_AST = (AST)currentAST.r
oot;
break;
}
case LPAREN:
{
{
match(LPAREN);
expression();
astFactory.addASTChild(currentAST
, returnAST);
match(RPAREN);
}
primitive_AST = (AST)currentAST.r
oot;
```

```
break;
}
case LITERAL_call:
{
AST tmp18_AST = null;
tmp18_AST = astFactory.create(LT(
1));
astFactory.makeASTRoot(currentAST
, tmp18_AST);
match(LITERAL_call);
AST tmp19_AST = null;
tmp19_AST = astFactory.create(LT(
1));
astFactory.addASTChild(currentAST
, tmp19_AST);
match(FN_NAME);
match(LPAREN);
{
switch ( LA(1)) {
case LPAREN:
case StrConstant:
case IntConstant:
case LITERAL_call:
case Identifier:
case LNOT:
case LCURLY:
{
expression();
astFactory.addASTChild(cu
rrentAST, returnAST);
{
_loop17:
do {
if ((LA(1)==COMMA
)) {
match(COM
MA);
expressio
n();
astFactor
y.addASTChild(currentAST, returnAST);
}
else {
break _lo
op17;
}
} while (true);
}
break;
}
case RPAREN:
{
break;
}
default:
{
throw new NoViableAltExce
ption(LT(1), getFilename());
}
}
}
match(RPAREN);
primitive_AST = (AST)currentAST.r
oot;
break;
}
default:
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_5);
}
returnAST = primitive_AST;
}

public final void expression() throws RecognitionExceptio
n, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST expression_AST = null;

try {      // for error handling
switch ( LA(1)) {
case LPAREN:
```

```
case StrConstant:
case IntConstant:
case LITERAL_call:
case Identifier:
case LNOT:
{
concatExpr();
astFactory.addASTChild(currentAST
, returnAST);
{
_loop50:
do {
if ((LA(1)==REGEXP)) {
{
AST tmp23_AST = n
ull;
tmp23_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp23_AST);
match(REGEXP);
}
concatExpr();
astFactory.addAST
Child(currentAST, returnAST);
}
else {
break _loop50;
}

} while (true);
}
expression_AST = (AST)currentAST.
root;
break;
}
case LCURLY:
{
AST tmp24_AST = null;
tmp24_AST = astFactory.create(LT(
1));
astFactory.makeASTRoot(currentAST
, tmp24_AST);
match(LCURLY);
{
switch ( LA(1)) {
```

```
case LPAREN:
case StrConstant:
case IntConstant:
case LITERAL_call:
case Identifier:
case LNOT:
case LCURLY:
{
expression();
astFactory.addASTChild(cu
rrentAST, returnAST);
{
_loop53:
do {
if ((LA(1)==COMMA
)) {
match(COM
MA);
expressio
n();
astFactor
y.addASTChild(currentAST, returnAST);
}
else {
break _lo
op53;
}

} while (true);
}
break;
}
case RCURLY:
{
break;
}
default:
{
throw new NoViableAltExce
ption(LT(1), getFilename());
}
}
}
match(RCURLY);
expression_AST = (AST)currentAST.
root;
```

```
break;
}
default:
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_6);
}
returnAST = expression_AST;
}

public final void unaryExpression() throws RecognitionExc
eption, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST unaryExpression_AST = null;

try {      // for error handling
{
{
_loop26:
do {
if ((LA(1)==LNOT)) {
AST tmp27_AST = null;
tmp27_AST = astFactory.cr
eate(LT(1));
astFactory.makeASTRoot(cu
rrentAST, tmp27_AST);
match(LNOT);
}
else {
break _loop26;
}

} while (true);
}
primitive();
astFactory.addASTChild(currentAST, return
AST);
```

```
}
unaryExpression_AST = (AST)currentAST.roo
t;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_5);
}
returnAST = unaryExpression_AST;
}

public final void multExpr() throws RecognitionException,
 TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST multExpr_AST = null;

try {      // for error handling
unaryExpression();
astFactory.addASTChild(currentAST, return
AST);
{
_loop30:
do {
if (((LA(1) >= STAR && LA(1) <= M
OD))) {
{
switch ( LA(1)) {
case STAR:
{
AST tmp28_AST = n
ull;
tmp28_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp28_AST);
match(STAR);
break;
}
case DIV:
{
AST tmp29_AST = n
ull;
tmp29_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp29_AST);
match(DIV);
break;
}
case MOD:
{
AST tmp30_AST = n
ull;
tmp30_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp30_AST);
match(MOD);
break;
}
default:
{
throw new NoViabl
eAltException(LT(1), getFilename());
}
}
}
unaryExpression();
astFactory.addASTChild(cu
rrentAST, returnAST);
}
else {
break _loop30;
}

} while (true);
}
multExpr_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_7);
}
returnAST = multExpr_AST;
}

public final void addExpr() throws RecognitionException,
TokenStreamException {
```

```
returnAST = null;
ASTPair currentAST = new ASTPair();
AST addExpr_AST = null;

try {      // for error handling
multExpr();
astFactory.addASTChild(currentAST, return
AST);
{
_loop34:
do {
if ((LA(1)==PLUS||LA(1)==MINUS))
{
{
switch ( LA(1)) {
case PLUS:
{
AST tmp31_AST = n
ull;
tmp31_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp31_AST);
match(PLUS);
break;
}
case MINUS:
{
AST tmp32_AST = n
ull;
tmp32_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp32_AST);
match(MINUS);
break;
}
default:
{
throw new NoViabl
eAltException(LT(1), getFilename());
}
}
}
multExpr();
```

```
astFactory.addASTChild(cu
rrentAST, returnAST);
}
else {
break _loop34;
}

} while (true);
}
addExpr_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_8);
}
returnAST = addExpr_AST;
}

public final void relExpr() throws RecognitionException,
TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST relExpr_AST = null;

try {      // for error handling
addExpr();
astFactory.addASTChild(currentAST, return
AST);
{
_loop38:
do {
if (((LA(1) >= EQUAL && LA(1) <=
LE))) {
{
switch ( LA(1)) {
case EQUAL:
{
AST tmp33_AST = n
ull;
tmp33_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp33_AST);
match(EQUAL);
break;
}
case NOT_EQUAL:
{
AST tmp34_AST = n
ull;
tmp34_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp34_AST);
match(NOT_EQUAL);
break;
}
case GT:
{
AST tmp35_AST = n
ull;
tmp35_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp35_AST);
match(GT);
break;
}
case GE:
{
AST tmp36_AST = n
ull;
tmp36_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp36_AST);
match(GE);
break;
}
case LT:
{
AST tmp37_AST = n
ull;
tmp37_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp37_AST);
match(LT);
break;
}
```

```
case LE:
{
AST tmp38_AST = n
ull;
tmp38_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp38_AST);
match(LE);
break;
}
default:
{
throw new NoViabl
eAltException(LT(1), getFilename());
}
}
}
addExpr();
astFactory.addASTChild(cu
rrentAST, returnAST);
}
else {
break _loop38;
}

} while (true);
}
relExpr_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_9);
}
returnAST = relExpr_AST;
}

public final void boolExpr() throws RecognitionException,
 TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST boolExpr_AST = null;

try {      // for error handling
```

```
relExpr();
astFactory.addASTChild(currentAST, return
AST);
{
_loop42:
do {
if ((LA(1)==LOR||LA(1)==LAND)) {
{
switch ( LA(1)) {
case LOR:
{
AST tmp39_AST = n
ull;
tmp39_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp39_AST);
match(LOR);
break;
}
case LAND:
{
AST tmp40_AST = n
ull;
tmp40_AST = astFa
ctory.create(LT(1));
astFactory.makeAS
TRoot(currentAST, tmp40_AST);
match(LAND);
break;
}
default:
{
throw new NoViabl
eAltException(LT(1), getFilename());
}
}
}
relExpr();
astFactory.addASTChild(cu
rrentAST, returnAST);
}
else {
break _loop42;
}
} while (true);
}
boolExpr_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_10);
}
returnAST = boolExpr_AST;
}

public final void concatExpr() throws RecognitionExceptio
n, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST concatExpr_AST = null;

try {      // for error handling
boolExpr();
astFactory.addASTChild(currentAST, return
AST);
{
_loop46:
do {
if ((LA(1)==DOT)) {
{
AST tmp41_AST = null;
tmp41_AST = astFactory.cr
eate(LT(1));
astFactory.makeASTRoot(cu
rrentAST, tmp41_AST);
match(DOT);
}
boolExpr();
astFactory.addASTChild(cu
rrentAST, returnAST);
}
else {
break _loop46;
}

} while (true);
}
concatExpr_AST = (AST)currentAST.root;
```

```
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_11);
}
returnAST = concatExpr_AST;
}

public final void assignstmnt() throws RecognitionExcepti
on, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST assignstmnt_AST = null;

try {      // for error handling
id();
astFactory.addASTChild(currentAST, return
AST);
{
switch ( LA(1)) {
case ASSIGN:
{
AST tmp42_AST = null;
tmp42_AST = astFactory.create(LT(
1));
astFactory.makeASTRoot(currentAST
, tmp42_AST);
match(ASSIGN);
break;
}
case HTTPGET:
{
AST tmp43_AST = null;
tmp43_AST = astFactory.create(LT(
1));
astFactory.makeASTRoot(currentAST
, tmp43_AST);
match(HTTPGET);
break;
}
default:
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
expression();
astFactory.addASTChild(currentAST, return
AST);
match(SEMI);
assignstmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = assignstmnt_AST;
}

public final void foreachstmnt() throws RecognitionExcept
ion, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST foreachstmnt_AST = null;

try {      // for error handling
AST tmp45_AST = null;
tmp45_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp45_
AST);
match(LITERAL_foreach);
AST tmp46_AST = null;
tmp46_AST = astFactory.create(LT(1));
astFactory.addASTChild(currentAST, tmp46_
AST);
match(Identifier);
match(LITERAL_in);
AST tmp48_AST = null;
tmp48_AST = astFactory.create(LT(1));
astFactory.addASTChild(currentAST, tmp48_
AST);
match(Identifier);
statementList();
astFactory.addASTChild(currentAST, return
AST);
foreachstmnt_AST = (AST)currentAST.root;
}
```

```
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = foreachstmnt_AST;
}

public final void whilestmnt() throws RecognitionExceptio
n, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST whilestmnt_AST = null;

try {      // for error handling
AST tmp49_AST = null;
tmp49_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp49_
AST);
match(LITERAL_while);
match(LPAREN);
expression();
astFactory.addASTChild(currentAST, return
AST);
match(RPAREN);
statementList();
astFactory.addASTChild(currentAST, return
AST);
whilestmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = whilestmnt_AST;
}

public final void ifstmnt() throws RecognitionException,
TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST ifstmnt_AST = null;
```

```
try {      // for error handling
AST tmp52_AST = null;
tmp52_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp52_
AST);
match(LITERAL_if);
match(LPAREN);
expression();
astFactory.addASTChild(currentAST, return
AST);
match(RPAREN);
statementList();
astFactory.addASTChild(currentAST, return
AST);
{
if ((LA(1)==LITERAL_else) && (_tokenSet_1
2.member(LA(2))) && (_tokenSet_13.member(LA(3)))) {
match(LITERAL_else);
statementList();
astFactory.addASTChild(currentAST
, returnAST);
}
else if ((_tokenSet_3.member(LA(1))) && (
_tokenSet_14.member(LA(2))) && (_tokenSet_15.member(LA(3)))) {
}
else {
throw new NoViableAltException(LT
(1), getFilename());
}

}
ifstmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = ifstmnt_AST;
}

public final void echostmnt() throws RecognitionException
, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
```

```
AST echostmnt_AST = null;

try {      // for error handling
AST tmp56_AST = null;
tmp56_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp56_
AST);
match(LITERAL_echo);
expression();
astFactory.addASTChild(currentAST, return
AST);
match(SEMI);
echostmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = echostmnt_AST;
}

public final void functionstmnt() throws RecognitionExcep
tion, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST functionstmnt_AST = null;

try {      // for error handling
AST tmp58_AST = null;
tmp58_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp58_
AST);
match(LITERAL_function);
AST tmp59_AST = null;
tmp59_AST = astFactory.create(LT(1));
astFactory.addASTChild(currentAST, tmp59_
AST);
match(FN_NAME);
match(LPAREN);
{
switch ( LA(1)) {
case Identifier:
{
id();
astFactory.addASTChild(currentAST
, returnAST);
{
_loop76:
do {
if ((LA(1)==COMMA)) {
match(COMMA);
id();
astFactory.addAST
Child(currentAST, returnAST);
}
else {
break _loop76;
}

} while (true);
}
break;
}
case RPAREN:
{
break;
}
default:
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
match(RPAREN);
fn_stmntlist();
astFactory.addASTChild(currentAST, return
AST);
functionstmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = functionstmnt_AST;
}

public final void includestmnt() throws RecognitionExcept
ion, TokenStreamException {
```

```
returnAST = null;
ASTPair currentAST = new ASTPair();
AST includestmnt_AST = null;

try {      // for error handling
AST tmp63_AST = null;
tmp63_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp63_
AST);
match(LITERAL_include);
match(LPAREN);
expression();
astFactory.addASTChild(currentAST, return
AST);
match(RPAREN);
match(SEMI);
includestmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = includestmnt_AST;
}

public final void returnstmnt() throws RecognitionExcepti
on, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST returnstmnt_AST = null;

try {      // for error handling
AST tmp67_AST = null;
tmp67_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp67_
AST);
match(LITERAL_return);
{
switch ( LA(1)) {
case LPAREN:
case StrConstant:
case IntConstant:
case LITERAL_call:
```

```
case Identifier:
case LNOT:
case LCURLY:
{
expression();
astFactory.addASTChild(currentAST
, returnAST);
break;
}
case SEMI:
{
break;
}
default:
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
match(SEMI);
returnstmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = returnstmnt_AST;
}

public final void nullstmnt() throws RecognitionException
, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST nullstmnt_AST = null;

try {      // for error handling
match(SEMI);
nullstmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
```

```
}
returnAST = nullstmnt_AST;
}

public final void callstmnt() throws RecognitionException
, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST callstmnt_AST = null;

try {      // for error handling
AST tmp70_AST = null;
tmp70_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp70_
AST);
match(LITERAL_call);
AST tmp71_AST = null;
tmp71_AST = astFactory.create(LT(1));
astFactory.addASTChild(currentAST, tmp71_
AST);
match(FN_NAME);
match(LPAREN);
{
switch ( LA(1)) {
case LPAREN:
case StrConstant:
case IntConstant:
case LITERAL_call:
case Identifier:
case LNOT:
case LCURLY:
{
expression();
astFactory.addASTChild(currentAST
, returnAST);
{
_loop58:
do {
if ((LA(1)==COMMA)) {
match(COMMA);
expression();
astFactory.addAST
Child(currentAST, returnAST);
}
else {
```

```
break _loop58;
}

} while (true);
}
break;
}
case RPAREN:
{
break;
}
default:
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
match(RPAREN);
match(SEMI);
callstmnt_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = callstmnt_AST;
}

public final void statementList() throws RecognitionExcep
tion, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST statementList_AST = null;

try {      // for error handling
switch ( LA(1)) {
case LCURLY:
{
AST tmp76_AST = null;
tmp76_AST = astFactory.create(LT(
1));
astFactory.makeASTRoot(currentAST
, tmp76_AST);
```

```
match(LCURLY);
{
_loop62:
do {
if ((_tokenSet_0.member(L
A(1)))) {
statement();
astFactory.addAST
Child(currentAST, returnAST);
}
else {
break _loop62;
}

} while (true);
}
AST tmp77_AST = null;
tmp77_AST = astFactory.create(LT(
1));
astFactory.addASTChild(currentAST
, tmp77_AST);
match(RCURLY);
statementList_AST = (AST)currentA
ST.root;
break;
}
case SEMI:
case LITERAL_call:
case Identifier:
case Perlstmnt:
case LITERAL_foreach:
case LITERAL_while:
case LITERAL_if:
case LITERAL_echo:
case LITERAL_function:
case LITERAL_include:
case LITERAL_return:
{
statement();
astFactory.addASTChild(currentAST
, returnAST);
statementList_AST = (AST)currentA
ST.root;
break;
}
default:
```

```
{
throw new NoViableAltException(LT
(1), getFilename());
}
}
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = statementList_AST;
}

public final void fn_stmntlist() throws RecognitionExcept
ion, TokenStreamException {

returnAST = null;
ASTPair currentAST = new ASTPair();
AST fn_stmntlist_AST = null;

try {      // for error handling
AST tmp78_AST = null;
tmp78_AST = astFactory.create(LT(1));
astFactory.makeASTRoot(currentAST, tmp78_
AST);
match(LCURLY);
{
_loop65:
do {
if ((_tokenSet_0.member(LA(1))))
{
statement();
astFactory.addASTChild(cu
rrentAST, returnAST);
}
else {
break _loop65;
}

} while (true);
}
AST tmp79_AST = null;
tmp79_AST = astFactory.create(LT(1));
astFactory.addASTChild(currentAST, tmp79_
AST);
```

```
match(RCURLY);
fn_stmntlist_AST = (AST)currentAST.root;
}
catch (RecognitionException ex) {
reportError(ex);
consume();
consumeUntil(_tokenSet_3);
}
returnAST = fn_stmntlist_AST;
}


public static final String[] _tokenNames = {
"<0>",
"EOF",
"<2>",
"NULL_TREE_LOOKAHEAD",
"\"begin\"",
"\"end\"",
"\"declare\"",
"FN_NAME",
"LPAREN",
"COMMA",
"RPAREN",
"SEMI",
"StrConstant",
"IntConstant",
"\"call\"",
"Identifier",
"LBRACK",
"RBRACK",
"LNOT",
"STAR",
"DIV",
"MOD",
"PLUS",
"MINUS",
"EQUAL",
"NOT_EQUAL",
"GT",
"GE",
"LT",
"LE",
"LOR",
"LAND",
"DOT",
"REGEXP",
"LCURLY",
"RCURLY",
"Perlstmnt",
"ASSIGN",
"HTTPGET",
"\"foreach\"",
"\"in\"",
"\"while\"",
"\"if\"",
"\"else\"",
"\"echo\"",
"\"function\"",
"\"include\"",
"\"return\"",
"LPERL",
"RPERL",
"ESC",
"Comment",
"VOCAB",
"Letter",
"Digit",
"Underscore",
"LineTerminator",
"WS"
};

protected void buildTokenTypeASTClassMap() {
tokenTypeToASTClassMap=null;
};

private static final long[] mk_tokenSet_0() {
long[] data = { 271098335774720L, 0L};
return data;
}
public static final BitSet _tokenSet_0 = new BitSet(mk_to
kenSet_0());
private static final long[] mk_tokenSet_1() {
long[] data = { 2L, 0L};
return data;
}
public static final BitSet _tokenSet_1 = new BitSet(mk_to
kenSet_1());
private static final long[] mk_tokenSet_2() {
long[] data = { 271098335774816L, 0L};
return data;
```

```
}
public static final BitSet _tokenSet_2 = new BitSet(mk_to
kenSet_2());
private static final long[] mk_tokenSet_3() {
long[] data = { 279928788535328L, 0L};
return data;
}
public static final BitSet _tokenSet_3 = new BitSet(mk_to
kenSet_3());
private static final long[] mk_tokenSet_4() {
long[] data = { 463856078336L, 0L};
return data;
}
public static final BitSet _tokenSet_4 = new BitSet(mk_to
kenSet_4());
private static final long[] mk_tokenSet_5() {
long[] data = { 51539217920L, 0L};
return data;
}
public static final BitSet _tokenSet_5 = new BitSet(mk_to
kenSet_5());
private static final long[] mk_tokenSet_6() {
long[] data = { 34359873024L, 0L};
return data;
}
public static final BitSet _tokenSet_6 = new BitSet(mk_to
kenSet_6());
private static final long[] mk_tokenSet_7() {
long[] data = { 51535547904L, 0L};
return data;
}
public static final BitSet _tokenSet_7 = new BitSet(mk_to
kenSet_7());
private static final long[] mk_tokenSet_8() {
long[] data = { 51522964992L, 0L};
return data;
}
public static final BitSet _tokenSet_8 = new BitSet(mk_to
kenSet_8());
private static final long[] mk_tokenSet_9() {
long[] data = { 50466000384L, 0L};
return data;
}
public static final BitSet _tokenSet_9 = new BitSet(mk_to
kenSet_9());
private static final long[] mk_tokenSet_10() {
```

```
long[] data = { 47244774912L, 0L};
return data;
}
public static final BitSet _tokenSet_10 = new BitSet(mk_t
okenSet_10());
private static final long[] mk_tokenSet_11() {
long[] data = { 42949807616L, 0L};
return data;
}
public static final BitSet _tokenSet_11 = new BitSet(mk_t
okenSet_11());
private static final long[] mk_tokenSet_12() {
long[] data = { 271115515643904L, 0L};
return data;
}
public static final BitSet _tokenSet_12 = new BitSet(mk_t
okenSet_12());
private static final long[] mk_tokenSet_13() {
long[] data = { 280358285605280L, 0L};
return data;
}
public static final BitSet _tokenSet_13 = new BitSet(mk_t
okenSet_13());
private static final long[] mk_tokenSet_14() {
long[] data = { 280358285605282L, 0L};
return data;
}
public static final BitSet _tokenSet_14 = new BitSet(mk_t
okenSet_14());
private static final long[] mk_tokenSet_15() {
long[] data = { 281474976577954L, 0L};
return data;
}
public static final BitSet _tokenSet_15 = new BitSet(mk_t
okenSet_15());

}
```

11.1.3 L.java

```
// $ANTLR 2.7.2: "20030511-debugging.g" -> "L.java"$

import java.io.InputStream;
import antlr.TokenStreamException;
import antlr.TokenStreamIOException;
import antlr.TokenStreamRecognitionException;
import antlr.CharStreamException;
import antlr.CharStreamIOException;
import antlr.ANTLRException;
import java.io.Reader;
import java.util.Hashtable;
import antlr.CharScanner;
import antlr.InputBuffer;
import antlr.ByteBuffer;
import antlr.CharBuffer;
import antlr.Token;
import antlr.CommonToken;
import antlr.RecognitionException;
import antlr.NoViableAltForCharException;
import antlr.MismatchedCharException;
import antlr.TokenStream;
import antlr.ANTLRHashString;
import antlr.LexerSharedInputState;
import antlr.collections.impl.BitSet;
import antlr.SemanticException;

public class L extends antlr.CharScanner implements PTokenTypes,
TokenStream
 {
public L(InputStream in) {
this(new ByteBuffer(in));
}
public L(Reader in) {
this(new CharBuffer(in));
}
public L(InputBuffer ib) {
this(new LexerSharedInputState(ib));
}
public L(LexerSharedInputState state) {
super(state);
caseSensitiveLiterals = true;
setCaseSensitive(true);
literals = new Hashtable();
literals.put(new ANTLRHashString("if", this), new Integer
(42));
literals.put(new ANTLRHashString("call", this), new Integ
er(14));
literals.put(new ANTLRHashString("function", this), new I
nteger(45));
literals.put(new ANTLRHashString("while", this), new Inte
ger(41));
literals.put(new ANTLRHashString("declare", this), new In
teger(6));
literals.put(new ANTLRHashString("end", this), new Intege
r(5));
literals.put(new ANTLRHashString("in", this), new Integer
(40));
literals.put(new ANTLRHashString("else", this), new Integ
er(43));
literals.put(new ANTLRHashString("include", this), new In
teger(46));
literals.put(new ANTLRHashString("begin", this), new Inte
ger(4));
literals.put(new ANTLRHashString("return", this), new Int
eger(47));
literals.put(new ANTLRHashString("echo", this), new Integ
er(44));
literals.put(new ANTLRHashString("foreach", this), new In
teger(39));
}

public Token nextToken() throws TokenStreamException {
Token theRetToken=null;
tryAgain:
for (;;) {
Token _token = null;
int _ttype = Token.INVALID_TYPE;
resetText();
try {   // for char stream error handling
try {   // for lexical error handling
switch ( LA(1)) {
case '(':
{
mLPAREN(true);
theRetToken=_returnToken;
break;
}
case ')':
{
mRPAREN(true);
```

```
theRetToken=_returnToken;
break;
}
case '[':
{
mLBRACK(true);
theRetToken=_returnToken;
break;
}
case ']':
{
mRBRACK(true);
theRetToken=_returnToken;
break;
}
case '{':
{
mLCURLY(true);
theRetToken=_returnToken;
break;
}
case ',':
{
mCOMMA(true);
theRetToken=_returnToken;
break;
}
case '.':
{
mDOT(true);
theRetToken=_returnToken;
break;
}
case ':':
{
mHTTPGET(true);
theRetToken=_returnToken;
break;
}
case '/':
{
mDIV(true);
theRetToken=_returnToken;
break;
}
case '+':
```

```
{
mPLUS(true);
theRetToken=_returnToken;
break;
}
case '-':
{
mMINUS(true);
theRetToken=_returnToken;
break;
}
case '*':
{
mSTAR(true);
theRetToken=_returnToken;
break;
}
case '%':
{
mMOD(true);
theRetToken=_returnToken;
break;
}
case '|':
{
mLOR(true);
theRetToken=_returnToken;
break;
}
case '@':
{
mREGEXP(true);
theRetToken=_returnToken;
break;
}
case ';':
{
mSEMI(true);
theRetToken=_returnToken;
break;
}
case '0':  case '1':  case '2':
case '3':
case '4':  case '5':  case '6':
case '7':
case '8':  case '9':
```

```
{
mIntConstant(true);
theRetToken=_returnToken;
break;
}
case '"':
{
mStrConstant(true);
theRetToken=_returnToken;
break;
}
case '#':
{
mComment(true);
theRetToken=_returnToken;
break;
}
case 'A':  case 'B':  case 'C':
case 'D':
case 'E':  case 'F':  case 'G':
case 'H':
case 'I':  case 'J':  case 'K':
case 'L':
case 'M':  case 'N':  case 'O':
case 'P':
case 'Q':  case 'R':  case 'S':
case 'T':
case 'U':  case 'V':  case 'W':
case 'X':
case 'Y':  case 'Z':  case 'a':
case 'b':
case 'c':  case 'd':  case 'e':
case 'f':
case 'g':  case 'h':  case 'i':
case 'j':
case 'k':  case 'l':  case 'm':
case 'n':
case 'o':  case 'p':  case 'q':
case 'r':
case 's':  case 't':  case 'u':
case 'v':
case 'w':  case 'x':  case 'y':
case 'z':
{
mFN_NAME(true);
theRetToken=_returnToken;
```

```
break;
}
case '$':
{
mIdentifier(true);
theRetToken=_returnToken;
break;
}
case '\t':  case '\n':  case '\r'
:  case ' ':
{
mWS(true);
theRetToken=_returnToken;
break;
}
default:
if ((LA(1)=='&') && (LA(2
)=='{') && ((LA(3) >= '\u0003' && LA(3) <= '\uffff'))) {
mPerlstmnt(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='=') &&
(LA(2)=='=')) {
mEQUAL(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='!') &&
(LA(2)=='=')) {
mNOT_EQUAL(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='>') &&
(LA(2)=='=')) {
mGE(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='<') &&
(LA(2)=='=')) {
mLE(true);
theRetToken=_retu
rnToken;
}
```

```
else if ((LA(1)=='&') &&
(LA(2)=='&')) {
mLAND(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='&') &&
(LA(2)=='{') && (true)) {
mLPERL(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='}') &&
(LA(2)=='&')) {
mRPERL(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='}') &&
(true)) {
mRCURLY(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='=') &&
(true)) {
mASSIGN(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='!') &&
(true)) {
mLNOT(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='>') &&
(true)) {
mGT(true);
theRetToken=_retu
rnToken;
}
else if ((LA(1)=='<') &&
(true)) {
mLT(true);
theRetToken=_retu
```

```
rnToken;
}
else {
if (LA(1)==EOF_CHAR) {upo
nEOF(); _returnToken = makeToken(Token.EOF_TYPE);}
else {throw new NoViableAltForCha
rException((char)LA(1), getFilename(), getLine(), getColumn());}
}
}
if ( _returnToken==null ) continu
e tryAgain; // found SKIP token
_ttype = _returnToken.getType();
_returnToken.setType(_ttype);
return _returnToken;
}
catch (RecognitionException e) {
throw new TokenStreamRecognitionE
xception(e);
}
}
catch (CharStreamException cse) {
if ( cse instanceof CharStreamIOException
 ) {
throw new TokenStreamIOException(
((CharStreamIOException)cse).io);
}
else {
throw new TokenStreamException(cs
e.getMessage());
}
}
}
}

public final void mLPAREN(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LPAREN;
int _saveIndex;

match('(');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mRPAREN(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = RPAREN;
int _saveIndex;

match(')');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mLBRACK(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LBRACK;
int _saveIndex;

match('[');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mRBRACK(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = RBRACK;
int _saveIndex;
```

```
match(']');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mLCURLY(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LCURLY;
int _saveIndex;

match('{');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mRCURLY(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = RCURLY;
int _saveIndex;

match('}');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mCOMMA(boolean _createToken) throws Rec
ognitionException, CharStreamException, TokenStreamException {
```

```
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = COMMA;
int _saveIndex;

match(',');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mDOT(boolean _createToken) throws Recog
nitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = DOT;
int _saveIndex;

match('.');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mASSIGN(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = ASSIGN;
int _saveIndex;

match('=');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
```

```
_returnToken = _token;
}

public final void mHTTPGET(boolean _createToken) throws R
ecognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = HTTPGET;
int _saveIndex;

match(":=");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mEQUAL(boolean _createToken) throws Rec
ognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = EQUAL;
int _saveIndex;

match("==");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mLNOT(boolean _createToken) throws Reco
gnitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LNOT;
int _saveIndex;

match('!');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mNOT_EQUAL(boolean _createToken) throws
 RecognitionException, CharStreamException, TokenStreamException
{
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = NOT_EQUAL;
int _saveIndex;

match("!=");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mDIV(boolean _createToken) throws Recog
nitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = DIV;
int _saveIndex;

match('/');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mPLUS(boolean _createToken) throws Reco
gnitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
```

```
ngth();
_ttype = PLUS;
int _saveIndex;

match('+');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mMINUS(boolean _createToken) throws Rec
ognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = MINUS;
int _saveIndex;

match('-');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mSTAR(boolean _createToken) throws Reco
gnitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = STAR;
int _saveIndex;

match('*');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
```

```
}

public final void mMOD(boolean _createToken) throws Recog
nitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = MOD;
int _saveIndex;

match('%');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mGE(boolean _createToken) throws Recogn
itionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = GE;
int _saveIndex;

match(">=");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mGT(boolean _createToken) throws Recogn
itionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = GT;
int _saveIndex;

match(">");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
```

```
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mLE(boolean _createToken) throws Recogn
itionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LE;
int _saveIndex;

match("<=");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mLT(boolean _createToken) throws Recogn
itionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LT;
int _saveIndex;

match('<');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mLOR(boolean _createToken) throws Recog
nitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LOR;
```

```
int _saveIndex;

match("||");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mLAND(boolean _createToken) throws Reco
gnitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LAND;
int _saveIndex;

match("&&");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mREGEXP(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = REGEXP;
int _saveIndex;

match('@');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}
```

```
public final void mSEMI(boolean _createToken) throws Reco
gnitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = SEMI;
int _saveIndex;

match(';');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mLPERL(boolean _createToken) throws Rec
ognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LPERL;
int _saveIndex;

match("&{");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mRPERL(boolean _createToken) throws Rec
ognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = RPERL;
int _saveIndex;

match("}&");
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mPerlstmnt(boolean _createToken) throws
 RecognitionException, CharStreamException, TokenStreamException
{
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = Perlstmnt;
int _saveIndex;

mLPERL(false);
{
_loop114:
do {
// nongreedy exit test
if ((LA(1)=='}') && (LA(2)=='&') && (true
)) break _loop114;
if (((LA(1) >= '\u0003' && LA(1) <= '\uff
ff')) && ((LA(2) >= '\u0003' && LA(2) <= '\uffff')) && ((LA(3) >=
 '\u0003' && LA(3) <= '\uffff'))) {
{
boolean synPredMatched112 = false
;
if (((LA(1)=='\r') && (LA(2)=='\n
') && ((LA(3) >= '\u0003' && LA(3) <= '\uffff')) && ((LA(4) >= '\
u0003' && LA(4) <= '\uffff')))) {
int _m112 = mark();
synPredMatched112 = true;
inputState.guessing++;
try {
{
match('\r');
match('\n');
}
}
catch (RecognitionExcepti
on pe) {
synPredMatched112
 = false;
}
rewind(_m112);
inputState.guessing--;
}
```

```
if ( synPredMatched112 ) {
match('\r');
match('\n');
if ( inputState.guessing=
=0 ) {
newline();
}
}
else if ((LA(1)=='\r') && ((LA(2)
 >= '\u0003' && LA(2) <= '\uffff')) && ((LA(3) >= '\u0003' && LA(
3) <= '\uffff')) && (true)) {
match('\r');
if ( inputState.guessing=
=0 ) {
newline();
}
}
else if ((LA(1)=='\n')) {
match('\n');
if ( inputState.guessing=
=0 ) {
newline();
}
}
else if ((_tokenSet_0.member(LA(1
)))) {
{
match(_tokenSet_0);
}
}
else {
throw new NoViableAltForC
harException((char)LA(1), getFilename(), getLine(), getColumn());
}

}
}
else {
break _loop114;
}

} while (true);
}
mRPERL(false);
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
```

```
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mIntConstant(boolean _createToken) thro
ws RecognitionException, CharStreamException, TokenStreamExceptio
n {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = IntConstant;
int _saveIndex;

{
{
int _cnt118=0;
_loop118:
do {
if (((LA(1) >= '0' && LA(1) <= '9'))) {
mDigit(false);
}
else {
if ( _cnt118>=1 ) { break _loop11
8; } else {throw new NoViableAltForCharException((char)LA(1), get
Filename(), getLine(), getColumn());}
}

_cnt118++;
} while (true);
}
}
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

protected final void mDigit(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
```

```
_ttype = Digit;
int _saveIndex;

{
{
matchRange('0','9');
}
}
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

protected final void mESC(boolean _createToken) throws Re
cognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = ESC;
int _saveIndex;

match('\\');
{
switch ( LA(1)) {
case 'n':
{
match('n');
break;
}
case 'r':
{
match('r');
break;
}
case 't':
{
match('t');
break;
}
case 'b':
{
match('b');
break;
}
case 'f':
{
match('f');
break;
}
case '"':
{
match('"');
break;
}
case '\'':
{
match('\'');
break;
}
case '\\':
{
match('\\');
break;
}
case '$':
{
match('$');
break;
}
case '.':
{
match('.');
break;
}
case '/':
{
match('/');
break;
}
case '(':
{
match('(');
break;
}
case ')':
{
match(')');
break;
}
```

```
case '?':
{
match('?');
break;
}
default:
{
throw new NoViableAltForCharException((ch
ar)LA(1), getFilename(), getLine(), getColumn());
}
}
}
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mStrConstant(boolean _createToken) thro
ws RecognitionException, CharStreamException, TokenStreamExceptio
n {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = StrConstant;
int _saveIndex;

_saveIndex=text.length();
match('"');
text.setLength(_saveIndex);
{
_loop124:
do {
if ((LA(1)=='\\')) {
mESC(false);
}
else if ((_tokenSet_1.member(LA(1)))) {
{
match(_tokenSet_1);
}
}
else {
break _loop124;
}
```

```
} while (true);
}
_saveIndex=text.length();
match('"');
text.setLength(_saveIndex);
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mComment(boolean _createToken) throws R
ecognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = Comment;
int _saveIndex;

{
{
match('#');
{
_loop130:
do {
if ((_tokenSet_0.member(LA(1)))) {
{
match(_tokenSet_0);
}
}
else {
break _loop130;
}

} while (true);
}
mLineTerminator(false);
}
}
if ( inputState.guessing==0 ) {
_ttype = Token.SKIP;
}
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

protected final void mLineTerminator(boolean _createToken
) throws RecognitionException, CharStreamException, TokenStreamEx
ception {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = LineTerminator;
int _saveIndex;

switch ( LA(1)) {
case '\r':
{
match('\r');
match('\n');
if ( inputState.guessing==0 ) {
newline();
}
break;
}
case '\n':
{
match('\n');
if ( inputState.guessing==0 ) {
newline();
}
break;
}
default:
{
throw new NoViableAltForCharException((ch
ar)LA(1), getFilename(), getLine(), getColumn());
}
}
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
```

```
_returnToken = _token;
}

protected final void mVOCAB(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = VOCAB;
int _saveIndex;

matchRange('\3','\377');
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mFN_NAME(boolean _createToken) throws R
ecognitionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = FN_NAME;
int _saveIndex;

{
mLetter(false);
}
{
_loop135:
do {
switch ( LA(1)) {
case 'A':  case 'B':  case 'C':  case 'D'
:
case 'E':  case 'F':  case 'G':  case 'H'
:
case 'I':  case 'J':  case 'K':  case 'L'
:
case 'M':  case 'N':  case 'O':  case 'P'
:
case 'Q':  case 'R':  case 'S':  case 'T'
:
case 'U':  case 'V':  case 'W':  case 'X'
:
```

```
case ’Y’:  case ’Z’:  case ’a’:  case ’b’
:
case ’c’:  case ’d’:  case ’e’:  case ’f’
:
case ’g’:  case ’h’:  case ’i’:  case ’j’
:
case ’k’:  case ’l’:  case ’m’:  case ’n’
:
case ’o’:  case ’p’:  case ’q’:  case ’r’
:
case ’s’:  case ’t’:  case ’u’:  case ’v’
:
case ’w’:  case ’x’:  case ’y’:  case ’z’
:
{
mLetter(false);
break;
}
case ’0’:  case ’1’:  case ’2’:  case ’3’
:
case ’4’:  case ’5’:  case ’6’:  case ’7’
:
case ’8’:  case ’9’:
{
mDigit(false);
break;
}
case ’_’:
{
mUnderscore(false);
break;
}
default:
{
break _loop135;
}
}
} while (true);
}
_ttype = testLiteralsTable(_ttype);
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
```

```
_returnToken = _token;
}

protected final void mLetter(boolean _createToken) throws
 RecognitionException, CharStreamException, TokenStreamException
{
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = Letter;
int _saveIndex;

{
switch ( LA(1)) {
case 'a':  case 'b':  case 'c':  case 'd':
case 'e':  case 'f':  case 'g':  case 'h':
case 'i':  case 'j':  case 'k':  case 'l':
case 'm':  case 'n':  case 'o':  case 'p':
case 'q':  case 'r':  case 's':  case 't':
case 'u':  case 'v':  case 'w':  case 'x':
case 'y':  case 'z':
{
{
matchRange('a','z');
}
break;
}
case 'A':  case 'B':  case 'C':  case 'D':
case 'E':  case 'F':  case 'G':  case 'H':
case 'I':  case 'J':  case 'K':  case 'L':
case 'M':  case 'N':  case 'O':  case 'P':
case 'Q':  case 'R':  case 'S':  case 'T':
case 'U':  case 'V':  case 'W':  case 'X':
case 'Y':  case 'Z':
{
{
matchRange('A','Z');
}
break;
}
default:
{
throw new NoViableAltForCharException((ch
ar)LA(1), getFilename(), getLine(), getColumn());
}
}
}
```

```
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

protected final void mUnderscore(boolean _createToken) th
rows RecognitionException, CharStreamException, TokenStreamExcept
ion {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = Underscore;
int _saveIndex;

{
{
match('_');
}
}
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mIdentifier(boolean _createToken) throw
s RecognitionException, CharStreamException, TokenStreamException
 {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = Identifier;
int _saveIndex;

{
{
match('$');
}
{
int _cnt140=0;
_loop140:
```

```
do {
switch ( LA(1)) {
case 'A':  case 'B':  case 'C':  case 'D'
:
case 'E':  case 'F':  case 'G':  case 'H'
:
case 'I':  case 'J':  case 'K':  case 'L'
:
case 'M':  case 'N':  case 'O':  case 'P'
:
case 'Q':  case 'R':  case 'S':  case 'T'
:
case 'U':  case 'V':  case 'W':  case 'X'
:
case 'Y':  case 'Z':  case 'a':  case 'b'
:
case 'c':  case 'd':  case 'e':  case 'f'
:
case 'g':  case 'h':  case 'i':  case 'j'
:
case 'k':  case 'l':  case 'm':  case 'n'
:
case 'o':  case 'p':  case 'q':  case 'r'
:
case 's':  case 't':  case 'u':  case 'v'
:
case 'w':  case 'x':  case 'y':  case 'z'
:
{
mLetter(false);
break;
}
case '0':  case '1':  case '2':  case '3'
:
case '4':  case '5':  case '6':  case '7'
:
case '8':  case '9':
{
mDigit(false);
break;
}
case '_':
{
mUnderscore(false);
break;
}
```

```
default:
{
if ( _cnt140>=1 ) { break _loop14
0; } else {throw new NoViableAltForCharException((char)LA(1), get
Filename(), getLine(), getColumn());}
}
}
_cnt140++;
} while (true);
}
}
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}

public final void mWS(boolean _createToken) throws Recogn
itionException, CharStreamException, TokenStreamException {
int _ttype; Token _token=null; int _begin=text.le
ngth();
_ttype = WS;
int _saveIndex;

{
int _cnt154=0;
_loop154:
do {
switch ( LA(1)) {
case ' ':
{
match(' ');
break;
}
case '\t':
{
match('\t');
break;
}
case '\n':  case '\r':
{
mLineTerminator(false);
break;
```

```
}
default:
{
if ( _cnt154>=1 ) { break _loop15
4; } else {throw new NoViableAltForCharException((char)LA(1), get
Filename(), getLine(), getColumn());}
}
}
_cnt154++;
} while (true);
}
if ( inputState.guessing==0 ) {
_ttype = Token.SKIP;
}
if ( _createToken && _token==null && _ttype!=Toke
n.SKIP ) {
_token = makeToken(_ttype);
_token.setText(new String(text.getBuffer(
), _begin, text.length()-_begin));
}
_returnToken = _token;
}


private static final long[] mk_tokenSet_0() {
long[] data = new long[2048];
data[0]=-9224L;
for (int i = 1; i<=1023; i++) { data[i]=-1L; }
return data;
}
public static final BitSet _tokenSet_0 = new BitSet(mk_to
kenSet_0());
private static final long[] mk_tokenSet_1() {
long[] data = new long[2048];
data[0]=-17179869192L;
data[1]=-268435457L;
for (int i = 2; i<=1023; i++) { data[i]=-1L; }
return data;
}
public static final BitSet _tokenSet_1 = new BitSet(mk_to
kenSet_1());

}
```

11.1.4 PTokenTypes.java

```
// $ANTLR 2.7.2: "20030511-debugging.g" -> "P.java"$

public interface PTokenTypes {
int EOF = 1;
int NULL_TREE_LOOKAHEAD = 3;
int LITERAL_begin = 4;
int LITERAL_end = 5;
int LITERAL_declare = 6;
int FN_NAME = 7;
int LPAREN = 8;
int COMMA = 9;
int RPAREN = 10;
int SEMI = 11;
int StrConstant = 12;
int IntConstant = 13;
int LITERAL_call = 14;
int Identifier = 15;
int LBRACK = 16;
int RBRACK = 17;
int LNOT = 18;
int STAR = 19;
int DIV = 20;
int MOD = 21;
int PLUS = 22;
int MINUS = 23;
int EQUAL = 24;
int NOT_EQUAL = 25;
int GT = 26;
int GE = 27;
int LT = 28;
int LE = 29;
int LOR = 30;
int LAND = 31;
int DOT = 32;
int REGEXP = 33;
int LCURLY = 34;
int RCURLY = 35;
int Perlstmnt = 36;
int ASSIGN = 37;
int HTTPGET = 38;
int LITERAL_foreach = 39;
int LITERAL_in = 40;
int LITERAL_while = 41;
int LITERAL_if = 42;
int LITERAL_else = 43;
int LITERAL_echo = 44;
int LITERAL_function = 45;
int LITERAL_include = 46;
int LITERAL_return = 47;
int LPERL = 48;
int RPERL = 49;
int ESC = 50;
int Comment = 51;
int VOCAB = 52;
int Letter = 53;
int Digit = 54;
int Underscore = 55;
int LineTerminator = 56;
int WS = 57;
}
```

11.1.5 PTokenTypes.txt

11.2 Semantic Checker

```
//Semantics.java
/**
 *
 *
 *@author=Omar Ahmed
 *
 *
 *
*/

import antlr.CommonAST;
import antlr.collections.AST;
import java.util.*;

public class Semantics
{

    private CommonAST tree;
    private Hashtable variables;
    private Vector functions;

    private int testWork;
    public Semantics(CommonAST c)
    {
testWork = 0;
try
    {
tree = c;
variables = new Hashtable();
functions = new Vector();
//checkSemantics();
    }
catch (Exception e)
    {
testWork = -1;
    }
    }

    public int checkSemantics()
    {
if (testWork == -1 || tree == null)
    {
System.err.println("Malformed tree, ERROR IN SEMA
NTIC CHECKING");
return -1;
    }
int childrenNumber = 0;
childrenNumber = tree.getNumberOfChildren();

int retVal = 0;
if (childrenNumber > 0)
    {
CommonAST childTree = (CommonAST)(tree.getFirstCh
ild());
retVal = statement (childTree.getText(), childTre
e);
if (retVal < 0)
    {
System.err.println("ERROR IN SEMANTIC CHE
CKING!");
return -1;
    }
childrenNumber--;
while (childrenNumber > 0)
    {
childTree = (CommonAST)(childTree.getNext
Sibling());
retVal = statement (childTree.getText(),
childTree);
if (retVal < 0)
    {
System.err.println("ERROR IN SEMA
NTIC CHECKING!");
return -1;
    }
childrenNumber--;
    }
    }
return 1;
    }

    //NEW DEVELOPMENT:
    //NO TYPES
    //just make sure things are defined or not.....return values,
 etc.
    //every time you put something into the hashtable, give it va
lue of 0

    public int statement (String nodeText, CommonAST node)
    {
```

```
if (nodeText.equals("if"))
    {
int numChildren = node.getNumberOfChildren();
if (numChildren != 2 && numChildren != 3)
    {
System.err.println("ill-formed if-else");
return -1;
    }
CommonAST ifClause = (CommonAST)node.getFirstChil
d();
int ifValue = firstlevelexpression(ifClause.getTe
xt(), ifClause);
if (ifValue < 0)
    {
System.err.println("error in if clause, u
nsupported condition");
return -1;
    }

CommonAST thenClause = (CommonAST)ifClause.getNex
tSibling();

if (thenClause.getText().equals("{"))
    {

CommonAST nextclause = (CommonAST)thenCla
use.getFirstChild();
if (nextclause.getText().equals("}"))
    {
    }
else
    {
while (!nextclause.getText().equa
ls( "}"))
    {
int nextvalue = statement
(nextclause.getText(), nextclause);
if (nextvalue < 0)
    {
System.err.printl
n("error in one of the statements in the 'then' clause of if-else
");
return -1;
    }
try
    {
```

```
nextclause = (Com
monAST)nextclause.getNextSibling();

    }
catch (Exception e)
    {
System.err.printl
n("problem in then clause of if-else");
return -1;
    }
    }
    }
    }
else
    {
int thenValue = statement(thenClause.getT
ext(), thenClause);
if (thenValue < 0)
    {
System.err.println("error, then c
lause of if-else is not valid");
return -1;
    }
    }

if (numChildren==3)
    {

CommonAST elseClause = null;
try
    {
elseClause = (CommonAST)thenClaus
e.getNextSibling();
    }
catch (Exception e)
    {
System.err.println("error trying
to get 'else' clause");
return -1;
    }
if (elseClause.getText().equals( "{"))
    {
CommonAST nextclause = null;
try
    {
nextclause = (CommonAST)e
```

```
lseClause.getFirstChild();
    }
catch (Exception e)
    {
System.err.println("error
 trying to get 'else' clause");
return -1;
    }
if (nextclause.getText().equals(
"}"))
    {
    }
else
    {
while (!nextclause.getTex
t().equals( "}"))
    {
int nextvalue = s
tatement(nextclause.getText(), nextclause);
if (nextvalue < 0
)
    {
System.er
r.println("error: invalid statement in 'else' clause");
return -1
;
    }
try
    {
nextclaus
e = (CommonAST)nextclause.getNextSibling();
    }
catch (Exception
e)
    {
System.er
r.println("error in 'else' clause");
return -1
;
    }
    }
    }

    }
else
    {
```

```
int elseValue = -1;
try
    {
elseValue = statement(els
eClause.getText(), elseClause);
    }
catch (Exception e)
    {
System.err.println("error
 in else clause");
return -1;
    }
if (elseValue < 0)
    {
System.err.println("error
: else clause is not a valid statement");
return -1;
    }
    }
    }
    }
else if (nodeText.equals("while"))
    {
//System.err.println("in while...");
if (node.getNumberOfChildren() != 2)
    {
System.err.println("while loop not proper
ly formed");
return -1;
    }
CommonAST conditionClause = (CommonAST)node.getFi
rstChild();
int conValue = firstlevelexpression (conditionCla
use.getText(), conditionClause);
if (conValue < 0)
    {
System.err.println("error, 'while' condit
ion is not valid expression");
return -1;
    }
CommonAST actionClause = (CommonAST)conditionClau
se.getNextSibling();
//System.err.println("first actionClause text = "
+actionClause.getText());
if (actionClause.getText().equals("{"))
    {
```

```
//System.err.println("first actionClause
text = "+actionClause.getText());
CommonAST nextclause = (CommonAST)actionC
lause.getFirstChild();
if (nextclause.getText().equals( "}"))
    {
    }
else
    {
while (!nextclause.getText().equa
ls( "}"))
    {
int nextvalue = statement
(nextclause.getText(), nextclause);
if (nextvalue < 0)
    {
System.err.printl
n("error with one of the statements in 'while' block");
return -1;
    }
try
    {
nextclause = (Com
monAST)nextclause.getNextSibling();
    }
catch (Exception e)
    {
System.err.printl
n("error in while block");
return -1;
    }
    }
    }


    }
else
    {
int actValue = statement (actionClause.ge
tText(), actionClause);
if (actValue < 0)
    {
System.err.println("error: while
action is not valid statement");
return -1;
    }
    }
    }
else if (nodeText.equals("foreach"))
    {
if (node.getNumberOfChildren() != 3)
    {
System.err.println("foreach not properly
formed");
return -1;
    }
//typeclasue and container clause should have NO
children
CommonAST typeClause = (CommonAST)node.getFirstCh
ild();
if (typeClause.getNumberOfChildren() > 0)
    {
System.err.println("error in foreach, aro
und "+typeClause.getText()+"; should be an identifer");
return -1;
    }
String typeClauseText = typeClause.getText();
if (variables.containsKey (typeClauseText))
    {
//System.err.println("contents of variabl
es are : ");
//System.err.println(variables.toString()
);
System.err.println("error in foreach, "+t
ypeClause.getText()+" should not be declared");
return -1;
    }
CommonAST containerClause = (CommonAST)typeClause
.getNextSibling();
if (containerClause.getNumberOfChildren() > 0)
    {
System.err.println("error in foreach, aro
und "+containerClause.getText()+"; should be an identifier");
return -1;
    }
//make sure that containerClause type is containe
r type (int array or string array)

if (!variables.containsKey(containerClause.getTex
t()))
    {
System.err.println("error: "+containerCla
use.getText()+" is not declared");
return -1;
    }
int z = ((Integer)variables.get(containerClause.g
etText())).intValue();

variables.put(typeClause.getText(), new Integer(0
));

if (!variables.containsKey (containerClause.getTe
xt()))
    {
//System.err.println("error in statement,
 line 283, text="+nodeText);
return -1;
    }

CommonAST doClause = (CommonAST)containerClause.g
etNextSibling();
if (doClause.getText().equals("{"))
    {
CommonAST nextclause = (CommonAST)doClaus
e.getFirstChild();
if (nextclause.getText().equals("}"))
    {
    }
else
    {
while (!nextclause.getText().equa
ls("}"))
    {
int nextvalue = statement
(nextclause.getText(), nextclause);
if (nextvalue < 0)
    {
System.err.printl
n("error with one of the statements in foreach block");
return -1;
    }
try
    {
nextclause = (Com
monAST)nextclause.getNextSibling();
    }
catch (Exception e)
    {
```

```
System.err.printl
n("error in foreach, malformed action block");
return -1;
    }
    }
    }

    }
else
    {
int doValue = statement (doClause.getText
(), doClause);
if (doValue < 0)
    {
//System.err.println("error: fore
ach action is not a valid statement");
return -1;
    }
    }

//remove typeClauseText
variables.remove(typeClauseText);
    }

//UPDATE THIS
else if (nodeText.equals("call"))
    {
// i.e. call sum($a, $b);

int num_params = (node.getNumberOfChildren()) - 1
;

CommonAST childNode = (CommonAST)node.getFirstChi
ld();
String functionName = childNode.getText();

int x = functions.size();

//find out how many params the function is suppos
ed to take
int numParamsNeeded = -1;

for (int i=0; i< x; i++)
    {
Vector v = (Vector)functions.elementAt(i)
;
```

```
String s = (String)v.elementAt(0);
if (s.equals(functionName))
    {
numParamsNeeded = v.size() - 1;
    }
    }

if (numParamsNeeded == -1)
    {
System.err.println("error, call with no f
unction name, or parameters");
return -1;
    }

if (numParamsNeeded != num_params)
    {
System.err.println("error, wrong number o
f parameters");
return -1;
    }

while (num_params > 0)
    {
childNode = (CommonAST)childNode.getNextS
ibling();
String paramName = childNode.getText();

int z = secondlevelexpression (paramName,
 childNode);
if (z < 0)
    {
System.err.println("error: invali
d parameter");
return -1;
    }
num_params--;
    }

    }
else if (nodeText.startsWith("&{"))
    {
return 0;
    }
//UPDATE THIS
else if (nodeText.equals("declare"))
    {
```

```
Vector params = new Vector();
int numChildren = node.getNumberOfChildren();
if (numChildren > 0)
    {
CommonAST childNode = (CommonAST)node.get
FirstChild();
if (childNode.getNumberOfChildren() > 0)
    {
System.err.println("error, malfor
med declare");
return -1;
    }
String functionName = childNode.getText()
;
params.add(functionName);

numChildren--;
while (numChildren > 0)
    {
childNode = (CommonAST)childNode.
getNextSibling();
String argType = childNode.getTex
t();
params.add(argType);
numChildren--;
    }

if (functions.contains(params))
    {
System.err.println("error, functi
on "+functionName+" already defined");
return -1;
    }
else
    {
functions.add(params);
    }
    }
else
    {
System.err.println("error: declare with n
o functin name or parameters");
return -1;
    }
    }
//UPDATE THIS
```

```
else if (nodeText.equals("function"))
    {
Vector params = new Vector();
int numChildren = node.getNumberOfChildren();
if (numChildren > 0)
    {
CommonAST childNode = (CommonAST)node.get
FirstChild();
if (childNode.getNumberOfChildren() > 0)
    {
System.err.println("error, malfor
med function");
return -1;
    }
String functionName = childNode.getText()
;
params.add(functionName);

numChildren--;
while (numChildren > 0)
    {
childNode = (CommonAST)childNode.
getNextSibling();
String argType = childNode.getTex
t();
if (numChildren != 1)
    params.add(argType);
numChildren--;
    }

//for (int j=0; j<params.size(); j++)
//  {
//System.err.println((String)para
ms.elementAt(j));
//  }
if (!functions.contains(params))
    {
System.err.println("error, functi
on "+functionName+" not defined");
return -1;
    }


//now the next node is the open brace...i
ts children are the statements
int numCh = childNode.getNumberOfChildren
```

```
();
if (numCh == 0)
    {
System.err.println("error, no fun
ction body");
return -1;
    }

for (int y=1; y < params.size(); y++)
    {
variables.put((String)params.elem
entAt(y), new Integer(0));
    }
CommonAST childChild = (CommonAST)childNo
de.getFirstChild();
String childChildText = childChild.getTex
t();
int x = statement (childChildText, childC
hild);
//System.err.println("first line");
if (x < 0)
    {
System.err.println("function "+fu
nctionName+" has an error");
//System.err.println("the root no
de here is "+childChildText);
return -1;
    }
numCh--;


while (numCh > 1)
    {
// > 1 and not > 0 because the la
st child is the }
childChild = (CommonAST)childChil
d.getNextSibling();
childChildText = childChild.getTe
xt();
x = statement (childChildText, ch
ildChild);
//System.err.println(" line : "+n
umCh);
if (x < 0)
    {
System.err.println("funct
ion "+functionName+" has an error");
//System.err.println("the
 root node here is "+childChildText);
return -1;
    }
numCh--;
    }

for (int y=1; y<params.size(); y++)
    {
variables.remove((String)params.e
lementAt(y));
    }

    }
else
    {
System.err.println("error; calling 'funct
ion' with no function name or params");
return -1;
    }
    }

//do i need to update this?
else if (nodeText.equals("return") || nodeText.equals("ec
ho"))
    {
//does this have children?  if so, they are expre
ssions

if (node.getNumberOfChildren() > 0)
    {
if (node.getNumberOfChildren() != 1)
    {
System.err.println("error, malfor
med "+nodeText);
return -1;
    }

CommonAST child = (CommonAST)node.getFirs
tChild();
String childText = child.getText();
int z = expression (childText, child);
if (z < 0)
    {
System.err.println("error: "+node
Text+" does not have a valid expression");
return -1;
    }
    }

return 0;
    }
else if (nodeText.equals("="))
    {
if (node.getNumberOfChildren() != 2)
    {
System.err.println("error; '=' needs an i
dentifier on one side and an expression on the other");
return -1;
    }
CommonAST leftSide = (CommonAST)node.getFirstChil
d();


//if rightSide has no children, then it is a stri
ng constant
CommonAST rightSide = (CommonAST)leftSide.getNext
Sibling();

if (rightSide.getText().equals("{"))
    {
//list
int x = rightSide.getNumberOfChildren();
CommonAST listElement = null;
if (x > 0)
    {
listElement = (CommonAST)rightSid
e.getFirstChild();
int z = expression (listElement.g
etText(), listElement);
if (z < 0)
    {
System.err.println("error
:  list element being assigned is invalid");
return -1;
    }
x--;
    }
while (x > 0)
    {
listElement = (CommonAST)listElem
```

```
ent.getNextSibling();
int z = expression (listElement.g
etText(), listElement);
if (x < 0)
    {
System.err.println("error
:  list element being assigned is invalid");
return -1;
    }
x--;
    }
    }
else
    {
int z = 0;

z = secondlevelexpression (rightSide.getT
ext(), rightSide);
if (z < 0)
    {
System.err.println("error:  inval
id right side of '=' expression");
return -1;
    }
    }


String s = leftSide.getText();
if (s.indexOf('[') > 0)
    {
s = s.substring(0, s.indexOf('['));
    }
if (!variables.containsKey(s))
    {
variables.put(s, new Integer(0));

    }
    }
else if (nodeText.equals(":="))
    {
//must be only two children

if (node.getNumberOfChildren() != 2)
    {
System.err.println("error;':=' takes iden
tifer on left, expression on right");
```

```
return -1;
    }
CommonAST leftSide = (CommonAST)node.getFirstChil
d();
//left side is an identifier


//right side could just be a string, if it has no
 children
CommonAST rightSide = (CommonAST)leftSide.getNext
Sibling();
int z = 0;

z = secondlevelexpression (rightSide.getText(), r
ightSide);

if (z < 0)
    {
System.err.println("error: invalid right
side of ':=' expression");
return -1;
    }
//both sides must be strings

String s = leftSide.getText();
int x = s.indexOf('[');
if (x > 0)
    {
s = s.substring(0, x);

if (!variables.containsKey(s))
    {
//System.err.println("putting ("+
s+","+2+")");
variables.put(s, new Integer(0));
    }
    }
else
    {
if (!variables.containsKey(s))
    {
//System.err.println("putting ("+
s+","+0+")");
variables.put(s, new Integer(0));
    }
    }
    }
else
    {
System.err.println("error, '"+nodeText+"' does no
t belong there");
return -1;
    }

return 0;
    }



    public int firstlevelexpression (String nodeText, CommonAST n
ode)
    {
if (nodeText.equals("==") || nodeText.equals("!=") || nod
eText.equals("<") || nodeText.equals(">") || nodeText.equals("<="
) || nodeText.equals(">=") || nodeText.equals("||") || nodeText.e
quals("&&"))
    {
if (node.getNumberOfChildren() != 2)
    {
System.err.println("error in '"+nodeText+
"' expression");
return -1;
    }
CommonAST leftSide = (CommonAST)node.getFirstChil
d();
int z1 = expression (leftSide.getText(), leftSide
);
if (z1 < 0)
    {
//System.err.println("error in firstlevel
expression, line 609, text="+nodeText);
return -1;
    }
CommonAST rightSide = (CommonAST)leftSide.getNext
Sibling();
int z2 = expression (rightSide.getText(), rightSi
de);
if (z2 < 0)
    {
//System.err.println("error in firstlevel
expression, line 616, text="+nodeText);
```

```
return -1;
    }
    }
else
    {
System.err.println("error; '"+nodeText+" does not
 evaluate an expression");
return -1;
    }
return 0;
    }

    public int secondlevelexpression (String nodeText, CommonAST
node)
    {
//start simple...

if (nodeText.equals(".") || nodeText.equals("@") || nodeT
ext.equals("+") || nodeText.equals("-") || nodeText.equals("*") |
| nodeText.equals("/"))
    {
//both sides must be strings or string constants

if (node.getNumberOfChildren() != 2)
    {
System.err.println("error, '"+nodeText+"'
 needs an expression, identifer, or constant on either side");
return -1;
    }
CommonAST firstchild = (CommonAST)node.getFirstCh
ild();
int checkfirst = secondlevelexpression (firstchil
d.getText(), firstchild);
if (checkfirst < 0)
    {
//System.err.println("error in secondleve
lexpression, line 646, text="+nodeText);
return -1;
    }

CommonAST secondchild = (CommonAST)firstchild.get
NextSibling();
int checksecond = secondlevelexpression (secondch
ild.getText(), secondchild);
if (checksecond < 0)
    {
```

```
//System.err.println("error in secondleve
lexpression, line 654, text="+nodeText);
return -1;
    }
return 0;

    }
else if (nodeText.equals("call"))
    {
//WAAAAAA!
// i.e. call sum($a, $b);

int num_params = (node.getNumberOfChildren()) - 1
;

CommonAST childNode = (CommonAST)node.getFirstChi
ld();
String functionName = childNode.getText();

int x = functions.size();

//find out how many params the function is suppos
ed to take
int numParamsNeeded = -1;

for (int i=0; i< x; i++)
    {
Vector v = (Vector)functions.elementAt(i)
;
String s = (String)v.elementAt(0);
if (s.equals(functionName))
    {
numParamsNeeded = v.size() - 1;
    }
    }

if (numParamsNeeded == -1)
    {
System.err.println("error, calling functi
on without giving function name");
return -1;
    }

if (numParamsNeeded != num_params)
    {
System.err.println("error, function "+fun
ctionName+" needs "+numParamsNeeded+" parameters");
return -1;
    }

while (num_params > 0)
    {
childNode = (CommonAST)childNode.getNextS
ibling();
String paramName = childNode.getText();

int z = secondlevelexpression (paramName,
 childNode);
if (z < 0)
    {
//System.err.println("error");
return -1;
    }
num_params--;
    }
return 0;
    }
else
    {
//here it can be a number, ID, or string constant

//if (node.getNumberOfChildren() > 0)
//    {
// System.err.println("error: '"+nodeText+"'
 is not an identifier or constant");
// return -1;
//    }
//ID

if (node.getNumberOfChildren() > 0)
    {
if (node.getNumberOfChildren() != 3)
    {
System.err.println("error:  ill-f
ormed array");
return -1;
    }
CommonAST arrayNode = (CommonAST)node.get
FirstChild();
if (!arrayNode.getText().equals("["))
    {
System.err.println("error:  this
is not a proper array");
return -1;
    }
CommonAST secondNode = (CommonAST)arrayNo
de.getNextSibling();
CommonAST endBrace = (CommonAST)secondNod
e.getNextSibling();
if (!endBrace.getText().equals("]"))
    {
System.err.println("error:  ill f
ormed array");
return -1;
    }

int w = expression (secondNode.getText(),
 secondNode);
if (w < 0)
    {
return -1;
    }

//proper array

    }
if (nodeText.indexOf('$') == 0)
    {
if (nodeText.indexOf('[') > 0)
    {
String s = nodeText.substring(0,
nodeText.indexOf('['));
if (!variables.containsKey(s))
    {
System.err.println("error
: the variable "+s+" has not been defined");
return -1;
    }
    }
else
    {
//it's a variable....get the type
 of the variable
if (!variables.containsKey(nodeTe
xt))
    {
System.err.println("error
: the variable "+nodeText+" has not been defined");
return -1;
    }
    }

//this should never be called
//return -1;
    }
return 0;
    }

//this should never be called, but just in case..
//return -1;
    }

    public int expression (String nodeText, CommonAST node)
    {
if (nodeText.equals("==") || nodeText.equals("!=") || nod
eText.equals("<") || nodeText.equals(">") || nodeText.equals("<="
) || nodeText.equals(">=") || nodeText.equals("||") || nodeText.e
quals("&&"))
    {
return firstlevelexpression (nodeText, node);
    }
return secondlevelexpression (nodeText, node);

    }

}
```

11.3 Code Generator

```
// Principal Author: Yuan Zheng

/*
 * CodeGenerator.java
 *
 * Created on May 3, 2003, 5:08 PM
 */

/**
 *
 * @author  Yuan Zheng
 */

import java.io.*;

import antlr.CommonAST;
import antlr.collections.AST;
import java.util.*;

public class CodeGenerator {

    /** privatevariables */
    private CommonAST tree; //the AST tree after static semantic
checking

    //use java's build in stack class in stead of the antlr's sta
ck interface
    private Stack stack;    //the stack to keep the generated cod
e
    private LinkedList codeList;
    private LinkedList httpVariableList; //keep track if a variab
le is a httpVariable
    //the temp file contains the html infor have to be removed at
 end
    private Hashtable scalarTable;  //use hashtable to keep track
 of all scalar variables
    private LinkedList arrayList;
    private int count;


    /** Creates a new instance of CodeGenerator */
    public CodeGenerator() {
        tree=null;
        stack = new Stack(); //stack might not be appropiate in y
```

```
olt code generation
        codeList = new LinkedList(); //list is better than stack?
?
        httpVariableList = new LinkedList();
        scalarTable = new Hashtable();
        arrayList = new LinkedList();
        count=0;
    }



    //constructor, takes in a CommonAST tree
    public CodeGenerator (CommonAST c){
        tree=c;
        stack = new Stack(); //stack might not be appropiate in y
olt code generation
        codeList = new LinkedList(); //list is better than stack?
?
        httpVariableList = new LinkedList();
        scalarTable = new Hashtable();
        arrayList = new LinkedList();
        count=0;
    }

    public LinkedList getCodeList(){
        return codeList;
    }

    public void result(){
        for (int i=0; i<codeList.size(); i++){
            System.out.println((String)codeList.get(i));
        }
    }

    public void generate(CommonAST s){
        int childrenNumber=0;
CommonAST ast=s;
        childrenNumber=s.getNumberOfChildren();
        //System.out.println("Number of child "+childrenNumber);
        if (childrenNumber>0){
            CommonAST childTree=(CommonAST)(s.getFirstChild());
            //System.out.println("child Tree: "+childTree.toStrin
gTree());
            condition (childTree.getText(), childTree);
            childrenNumber--;
            while (childrenNumber>0){
                childTree=(CommonAST)(childTree.getNextSibling())
;
                //System.out.println("child tree: "+childTree.toS
tringTree());
                condition (childTree.getText(), childTree);
                childrenNumber--;
            }
        }
    }


    public CommonAST getTree(){
return tree;
    }

    //method to find the right code generation method for the spe
cific root
    private void condition (String c, CommonAST s){
        CommonAST ast=s;
        String condition=c;
// System.out.println("Condition is "+condition);
if (condition==null){
    // System.out.println("nothing");
}
        else if (condition.equalsIgnoreCase(":=")){
            generateHTTPGET(ast);
            //done with the method
        }
        else if (condition.equalsIgnoreCase("=")){
            //System.out.println("assign root node "+ast.getText(
));
            generateAssign(ast);
        }
        else if (condition.equalsIgnoreCase("echo")){
            //System.out.println("echo root node "+ ast.getText()
);
            generateEcho(ast);
            //done with the method
        }
        //binary mathmematical oper
/*
        else if (condition.equalsIgnoreCase("+")){
            generatePlus(ast);  //return value???
        }
        else if (condition.equalsIgnoreCase("-")){
            generateMinus(ast);
        }
        else if (condition.equalsIgnoreCase("*")){
            generateStar(ast);
        }
        else if (condition.equalsIgnoreCase("/")){
            generateDivide(ast);
        }
        else if (condition.equalsIgnoreCase("%")){
        }
*/
        //end binary mathematical operations

//generate if statment, will never see an else
//in the AST root.  Parser gets rid of it
        else if (condition.equalsIgnoreCase("if")){
            generateIF(ast);
        }
else if (condition.equalsIgnoreCase("while")){
    generateWhile(ast);
}
else if (condition.equalsIgnoreCase("foreach")){
    generateForeach(ast);
        }
        else if (condition.equalsIgnoreCase("@")){
            generateAt(ast);
        }

        else if (condition.equalsIgnoreCase("declare")){
    //do nothing...
}
else if (condition.equalsIgnoreCase("function")){
    generateFun(ast);
}

else if (condition.equalsIgnoreCase("call")){
    //System.out.println("come here ever");
    generateCall(ast);
}
else if (condition.equalsIgnoreCase("return")){
    //System.out.println("ever come to return");
    generateReturn (ast);
}
else if (condition.charAt(0)=='&'){
    generatePerlBlock(ast);
    //perl block, need change in parser
        }
    }

    private String generateCall(CommonAST s){
//System.out.println("hello...i am sleepy");
CommonAST ast=s;
//how many args?
int argsCount=ast.getNumberOfChildren();
String code="";
ast=(CommonAST)ast.getFirstChild();
String name=ast.getText();
code+="&"+name;
if (argsCount>1){
    code+="(";
    //begin with args
    for (int i=1; i<argsCount; i++){
ast=(CommonAST)ast.getNextSibling();
String tmp=ast.getText();

if (tmp.charAt(0)=='$'){
    //do nothing;
}
else if (tmp.equalsIgnoreCase("+")){
    tmp=generatePlus(ast);
}
else if (tmp.equalsIgnoreCase("-")){
    tmp=generateMinus(ast);
}
else if (tmp.equalsIgnoreCase("*")){
    tmp=generateStar(ast);
}
else if (tmp.equalsIgnoreCase("/")){
    tmp=generateDivide(ast);
}
else if (tmp.equalsIgnoreCase("%")){
    tmp=generateMod(ast);
}
else if (tmp.charAt(0)<'9'&&tmp.charAt(0)>'0'){
    //do nothing;
}
else {
    tmp="\""+tmp+"\"";
}
code+=tmp+",";
    }
    //get rid of the last ,
    if (code.charAt(code.length()-1)==','){
```

```
code=code.substring(0, code.length()-1);
    }
    code+=");";
}
code+=";";
codeList.add(code);
return code;

    }

    private void generateReturn(CommonAST s){
CommonAST ast=s;
String code="return ";
ast=(CommonAST)ast.getFirstChild();
String text=ast.getText();
if (text.charAt(0)=='$'){
    //do nothing;
}
else if (text.equalsIgnoreCase("+")){
    text=generatePlus(ast);
}
else if (text.equalsIgnoreCase("-")){
    text=generateMinus(ast);
}
else if (text.equalsIgnoreCase("*")){
    text=generateStar(ast);
}
else if (text.equalsIgnoreCase("/")){
    text=generateDivide(ast);
}
else if (text.equalsIgnoreCase("%")){
    text=generateMod(ast);
}
else if (text.equalsIgnoreCase("call")){
    text=generateCall(ast);
    text=text.substring(0, text.length()-1);
}
code+=text+";";
codeList.add(code);
    }

    private void generateFun(CommonAST s){
CommonAST ast=s;
//one child for fun name, one for the code inside fun,
//one for the last }
int args=ast.getNumberOfChildren()-3;
```

```
ast=(CommonAST)ast.getFirstChild();
String name=ast.getText();
String code="";
code="sub "+name+"{";
codeList.add(code);
//replace all args with $_[number]
Hashtable funTable=new Hashtable();  //keep track the ord
er or args
for (int i=0; i<=args; i++){
    ast=(CommonAST)ast.getNextSibling();
    String key=ast.getText();
    //System.out.println("key is "+key);
    String value="$_["+i+"]";
    funTable.put(key, value);
}
//now, go throw the rest of the function tree, replace ar
gs
CommonAST funAST=(CommonAST)ast.getNextSibling();
//stem.out.println("first fun! "+funAST.toStringTree());
replace(funTable, funAST);
//System.out.println("second fun! "+funAST.toStringTree()
);

//after replace everything, generatecode
generate(funAST);
code="}";
codeList.add(code);
    }

    //replace is finally workiing
    private void replace (Hashtable t, CommonAST s){
CommonAST ast=s;
Hashtable funTable=t;
int n=ast.getNumberOfChildren();
//System.out.println("child number "+n);
if (n>0){
    ast=(CommonAST)ast.getFirstChild();
    String text=ast.getText();
    //System.out.println(text);
    if (text.charAt(0)=='$'){
if (funTable.containsKey(text)){
    //replace the text
    //System.out.println("replace");
    ast.setText((String)funTable.get(text));
}
    }
    else {
//go down to the tree
//System.out.println("should print");
replace(funTable, ast);
    }
    for (int i=0; i<n-1; i++){
ast=(CommonAST)ast.getNextSibling();
text=ast.getText();
if (text.charAt(0)=='$'){
    if (funTable.containsKey(text)){
//replace the text
ast.setText((String)funTable.get(text));
    }
}
else {
    //go down to the tree
    replace(funTable, ast);
}
    }

}
    }
    private String generateConcat(CommonAST s){
CommonAST ast=s;
String left, right;
ast=(CommonAST)ast.getFirstChild();

left=ast.getText();

if (left.equalsIgnoreCase(".")){
    left=generateConcat(ast);
}
else if (left.charAt(0)!='$'){
    left="\""+left+"\"";
}
else {

    left = generateIdentifier(ast);

    //do nothing
    // MARK: test to see if array
    //     if (left.indexOf('[') != -1) {
    // left.replace('$','@');
    //     }
}
ast=(CommonAST)ast.getNextSibling();
```

```
right=ast.getText();
if (right.equalsIgnoreCase(".")){
    right=generateConcat(ast);
}
else if (right.charAt(0)!='$'){
    right="\""+right+"\"";
}
else {

    right = generateIdentifier(ast);

    //do nothing;
    // MARK: test to see if array
    //     if (right.indexOf('[') != -1) {
    // right.replace('$','@');
    //     }
}
return left+"."+right;

    }


    //method to generate code for @ symbol
    //the regular expression matching, return the temp string
    private String generateAt(CommonAST s){
//put the result into a temp array first
        //should be only two child, the first one a scalar, and t
he second one an array
        String left, right;
        CommonAST ast=s;
        ast=(CommonAST)ast.getFirstChild();
        left=ast.getText();
        //left should be some regular expression
        ast=(CommonAST)ast.getNextSibling();
        right=ast.getText();
        right=right.replace('$',  '@');
        //begin generate perl code
        String code="";
        count++;
        String tmpArray="@tmp"+count;
        code=tmpArray+"=();";
codeList.add(code);
        code="foreach ( "+right+") {";
        codeList.add(code);
        //reset code
        code="if ($_=~m/("+left+")/i){";
        codeList.add(code);
code="push "+tmpArray+", $2}";
codeList.add(code);
code="}";
codeList.add(code);
return tmpArray;
    }

    private void generateForeach(CommonAST s){
CommonAST ast=s;
String code="";
code+="foreach ";
ast=(CommonAST)ast.getFirstChild();
code+=ast.getText() + " ";
ast=(CommonAST)ast.getNextSibling();
String array=ast.getText();
array=array.replace('$', '@');
code+="( " + array + " )";
//next go to the {} part
code+=" {";
codeList.add(code);
//stmt or stmt list
//then begin add whatever inside {}
CommonAST foreachPart=(CommonAST)ast.getNextSibling();
//with list of stmt
if (foreachPart.getText().equalsIgnoreCase("{")){
    generate(foreachPart);
}
else
    condition((foreachPart).getText(),foreachPart);

codeList.add("}");
    }

    //code for while statements
    private void generateWhile(CommonAST s){
CommonAST ast=s;
codeList.add("while");
CommonAST firstChild=(CommonAST)ast.getFirstChild();
String root=firstChild.getText();
if (root.equalsIgnoreCase("&&")){
    codeList.add(generateLogicAnd(firstChild));
}
else if (root.equalsIgnoreCase("||")){
    generateLogicOr(firstChild);
}
```

```
else if (root.equalsIgnoreCase("==")){
    String conditionCode;
    conditionCode=generateEQ(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase("!=")){
    String conditionCode;
    conditionCode=generateNE(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase(">=")){
    String conditionCode=generateGE(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase(">")){
    String conditionCode=generateGT(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase("<=")){
    String conditionCode=generateLE(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase("<")){
    String conditionCode=generateLT(firstChild);
    codeList.add(conditionCode);
}
else if (root.charAt(0)=='$'){
    //code to be added
    String conditionCode;
    conditionCode="( "+root+" )";
    codeList.add(conditionCode);
}
else {
    //shouldn't come at all
    System.out.println("Error:  Wrong conditions after if
");
    System.exit(1);
}

//stmt or stmt list
codeList.add("{");
//then begin add whatever inside {}
CommonAST whilePart=(CommonAST)firstChild.getNextSibling(
);
//with list of stmt
if (whilePart.getText().equalsIgnoreCase("{")){
    generate(whilePart);
}
else
    condition((whilePart).getText(),whilePart);

codeList.add("}");
    }


    //code for if statements
    private void generateIF (CommonAST s){
CommonAST ast=s;
int childNumber=ast.getNumberOfChildren();
String codeIf = "if";
codeList.add(codeIf);
//condition part for if is the first child
CommonAST firstChild=(CommonAST)ast.getFirstChild();
String root=firstChild.getText();
if (root.equalsIgnoreCase("&&")){
    codeList.add(generateLogicAnd(firstChild));
}
else if (root.equalsIgnoreCase("||")){
    codeList.add(generateLogicOr(firstChild));
}
else if (root.equalsIgnoreCase("==")){
    String conditionCode;
    conditionCode=generateEQ(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase("!=")){
    String conditionCode;
    conditionCode=generateNE(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase(">=")){
    String conditionCode=generateGE(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase(">")){
    String conditionCode=generateGT(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase("<=")){
    String conditionCode=generateLE(firstChild);
    codeList.add(conditionCode);
}
```

```
else if (root.equalsIgnoreCase("<")){
    String conditionCode=generateLT(firstChild);
    codeList.add(conditionCode);
}
else if (root.equalsIgnoreCase("!")){
    String conditionCode=generateNot(firstChild);
    codeList.add(conditionCode);
}
else if (root.charAt(0)=='$'){
    //code to be added
    String conditionCode;
    conditionCode="( "+root+" )";
    codeList.add(conditionCode);
}
else {
    //shouldn't come at all
    System.out.println("Error:  Wrong conditions after if
");
    System.exit(1);
}

//begin the part inside if {}
//add { first
codeList.add("{");
//then begin add whatever inside {}
CommonAST ifPart=(CommonAST)firstChild.getNextSibling();
//with list of stmt
if (ifPart.getText().equalsIgnoreCase("{")){
    generate(ifPart);
}
else
    condition((ifPart).getText(),ifPart);

codeList.add("}");

if (childNumber>2){
    //there is else part
    CommonAST elsePart=(CommonAST)ifPart.getNextSibling()
;
    codeList.add("else {");
    if (elsePart.getText().equalsIgnoreCase("{")){
generate(elsePart);
    }
    else
condition((elsePart).getText(), elsePart);
codeList.add("}");
```

```
//bugs here, no else if is implemented
}
    }

    private String generateNot(CommonAST s){
CommonAST ast=s;
ast=(CommonAST)ast.getFirstChild();
String code="(!";
code+=ast.getText()+")";
return code;
    }

    private String generateLogicAnd(CommonAST s){
String code="( ";
CommonAST ast=s;
ast=(CommonAST)ast.getFirstChild();
String left, right;
left=ast.getText();
CommonAST rightAST=(CommonAST)ast.getNextSibling();
right =ast.getNextSibling().getText();
//check the left part
if (left.equalsIgnoreCase("&&")){
    code+=generateLogicAnd(ast);
}
else if (left.equalsIgnoreCase("||")){
    code+=generateLogicOr(ast);
}
else if (left.equalsIgnoreCase("==")){
    code+=generateEQ(ast);
}
else if (left.equalsIgnoreCase("!=")){
    code+=generateNE(ast);
}
else if (left.equalsIgnoreCase(">=")){
    code+=generateGE(ast);
}
else if (left.equalsIgnoreCase(">")){
    code+=generateGT(ast);
}
else if (left.equalsIgnoreCase("<=")){
    code+=generateLE(ast);
}
else if (left.equalsIgnoreCase("<")){
    code+=generateLT(ast);
}
else if (left.charAt(0)=='$'){
    code+=left;
}
else {
    //shouldn't come at all
    System.out.println("Error:  Wrong conditions after if
");
    System.exit(1);
}
//add &&
code+=" && ";

//check the right part
if (right.equalsIgnoreCase("&&")){
    code+=generateLogicAnd(rightAST);
}
else if (right.equalsIgnoreCase("||")){
    code+=generateLogicOr(rightAST);
}
else if (right.equalsIgnoreCase("==")){
    code+=generateEQ(rightAST);
}
else if (right.equalsIgnoreCase("!=")){
    code+=generateNE(rightAST);
}
else if (right.equalsIgnoreCase(">=")){
    code+=generateGE(rightAST);
}
else if (right.equalsIgnoreCase(">")){
    code+=generateGT(rightAST);
}
else if (right.equalsIgnoreCase("<=")){
    code+=generateLE(rightAST);
}
else if (right.equalsIgnoreCase("<")){
    code+=generateLT(rightAST);
}
else if (right.charAt(0)=='$'){
    code+=right;
}
else {
    //shouldn't come at all
    System.out.println("Error:  Wrong conditions after if
");
    System.exit(1);
}
```

```
code+=" )";
return code;

    }

    private String generateLogicOr(CommonAST s){
String code="( ";
CommonAST ast=s;
ast=(CommonAST)ast.getFirstChild();
String left, right;
left=ast.getText();
CommonAST rightAST=(CommonAST)ast.getNextSibling();
right =ast.getNextSibling().getText();
//check the left part
if (left.equalsIgnoreCase("&&")){
    code+=generateLogicAnd(ast);
}
else if (left.equalsIgnoreCase("||")){
    code+=generateLogicOr(ast);
}
else if (left.equalsIgnoreCase("==")){
    code+=generateEQ(ast);
}
else if (left.equalsIgnoreCase("!=")){
    code+=generateNE(ast);
}
else if (left.equalsIgnoreCase(">=")){
    code+=generateGE(ast);
}
else if (left.equalsIgnoreCase(">")){
    code+=generateGT(ast);
}
else if (left.equalsIgnoreCase("<=")){
    code+=generateLE(ast);
}
else if (left.equalsIgnoreCase("<")){
    code+=generateLT(ast);
}
else if (left.charAt(0)=='$'){
    code+=left;
}
else {
    //shouldn't come at all
    System.out.println("Error:  Wrong conditions after if
");
    System.exit(1);
}
//add &&
code+=" || ";

//check the right part
if (right.equalsIgnoreCase("&&")){
    code+=generateLogicAnd(rightAST);
}
else if (right.equalsIgnoreCase("||")){
    code+=generateLogicOr(rightAST);
}
else if (right.equalsIgnoreCase("==")){
    code+=generateEQ(rightAST);
}
else if (right.equalsIgnoreCase("!=")){
    code+=generateNE(rightAST);
}
else if (right.equalsIgnoreCase(">=")){
    code+=generateGE(rightAST);
}
else if (right.equalsIgnoreCase(">")){
    code+=generateGT(rightAST);
}
else if (right.equalsIgnoreCase("<=")){
    code+=generateLE(rightAST);
}
else if (right.equalsIgnoreCase("<")){
    code+=generateLT(rightAST);
}
else if (right.charAt(0)=='$'){
    code+=right;
}
else {
    //shouldn't come at all
    System.out.println("Error:  Wrong conditions after if
");
    System.exit(1);
}

code+=" )";
return code;
    }

    /********************implement relational expressions********
***/
    /************>, <, ==, >=, <=, != **************************
```

```
**/
    /****************** all methods returns the code**********/

    //equal
    private String generateEQ(CommonAST s){
        CommonAST ast=s;
String left, right;
ast=(CommonAST)ast.getFirstChild();
left=ast.getText();
if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}
//no error checking here, need future improve
right=ast.getNextSibling().getText();
ast=(CommonAST)ast.getNextSibling();
if (right.charAt(0)=='$'){
    //do nothing;
}
else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
}
else if (right.equalsIgnoreCase("-")){
    right=generateMinus(ast);
}
else if (right.equalsIgnoreCase("*")){
    right=generateStar(ast);
}
else if (right.equalsIgnoreCase("/")){
    right=generateDivide(ast);
}
else if (right.equalsIgnoreCase("%")){
    right=generateMod(ast);
}
String code;
code="("+left+" eq "+right+")";
//codeList.add(code);
return code;
    }

    //not equal
    private String generateNE (CommonAST s) {
CommonAST ast=s;
String left, right;
ast=(CommonAST)ast.getFirstChild();
left=ast.getText();
//no error checking here, need future improve
if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}
right=ast.getNextSibling().getText();
ast=(CommonAST)ast.getNextSibling();
if (right.charAt(0)=='$'){
    //do nothing;
}
else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
}
else if (right.equalsIgnoreCase("-")){
    right=generateMinus(ast);
}
else if (right.equalsIgnoreCase("*")){
    right=generateStar(ast);
```

```
}
else if (right.equalsIgnoreCase("/")){
    right=generateDivide(ast);
}
else if (right.equalsIgnoreCase("%")){
    right=generateMod(ast);
}
String code;
code="("+left+" ne "+right+")";
//codeList.add(code);
return code;
    }

    //greater or equal
    private String generateGE(CommonAST s){
CommonAST ast=s;
String left, right;
ast=(CommonAST)ast.getFirstChild();
left=ast.getText();
//no error checking here, need future improve
if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}
right=ast.getNextSibling().getText();
ast=(CommonAST)ast.getNextSibling();
if (right.charAt(0)=='$'){
    //do nothing;
}
else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
}
```

```
else if (right.equalsIgnoreCase("-")){
    right=generateMinus(ast);
}
else if (right.equalsIgnoreCase("*")){
    right=generateStar(ast);
}
else if (right.equalsIgnoreCase("/")){
    right=generateDivide(ast);
}
else if (right.equalsIgnoreCase("%")){
    right=generateMod(ast);
}
String code;
code="("+left+" ge "+right+")";
//codeList.add(code);
return code;
    }

    //greater
    private String generateGT(CommonAST s){
CommonAST ast=s;
String left, right;
ast=(CommonAST)ast.getFirstChild();
left=ast.getText();
//no error checking here, need future improve
if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}
right=ast.getNextSibling().getText();
ast=(CommonAST)ast.getNextSibling();
if (right.charAt(0)=='$'){
    //do nothing;
}
else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
}
else if (right.equalsIgnoreCase("-")){
    right=generateMinus(ast);
}
else if (right.equalsIgnoreCase("*")){
    right=generateStar(ast);
}
else if (right.equalsIgnoreCase("/")){
    right=generateDivide(ast);
}
else if (right.equalsIgnoreCase("%")){
    right=generateMod(ast);
}
String code;
code="("+left+" gt "+right+")";
//codeList.add(code);
return code;
    }

    //less or equal
    private String generateLE(CommonAST s){
CommonAST ast=s;
String left, right;
ast=(CommonAST)ast.getFirstChild();
left=ast.getText();
//no error checking here, need future improve
if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}
right=ast.getNextSibling().getText();
ast=(CommonAST)ast.getNextSibling();
if (right.charAt(0)=='$'){
    //do nothing;
}
else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
}
else if (right.equalsIgnoreCase("-")){
    right=generateMinus(ast);
}
else if (right.equalsIgnoreCase("*")){
    right=generateStar(ast);
}
else if (right.equalsIgnoreCase("/")){
    right=generateDivide(ast);
}
else if (right.equalsIgnoreCase("%")){
    right=generateMod(ast);
}
String code;
code="("+left+" le "+right+")";
//codeList.add(code);
return code;
    }

    //less
    private String generateLT(CommonAST s){
CommonAST ast=s;
String left, right;
ast=(CommonAST)ast.getFirstChild();
left=ast.getText();
//no error checking here, need future improve
if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
```

```
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}
right=ast.getNextSibling().getText();
ast=(CommonAST)ast.getNextSibling();
if (right.charAt(0)=='$'){
    //do nothing;
}
else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
}
else if (right.equalsIgnoreCase("-")){
    right=generateMinus(ast);
}
else if (right.equalsIgnoreCase("*")){
    right=generateStar(ast);
}
else if (right.equalsIgnoreCase("/")){
    right=generateDivide(ast);
}
else if (right.equalsIgnoreCase("%")){
    right=generateMod(ast);
}
String code;
code="("+left+" lt "+right+")";
//codeList.add(code);
return code;
    }
    /***************End boolean relations operations here *******
*****/



    /***********implement binary mathematical operations*********
***/
    //all the mathematical operation + - * / % methods return the
 resulted int
    private String generatePlus(CommonAST s){
        CommonAST ast=s;
        String left, right;  //the left of +, and the right of +
        ast=(CommonAST)ast.getFirstChild();
        left= ast.getText();
if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}

        //now the right part of +
       ast=(CommonAST)ast.getNextSibling();
       right=ast.getText();
       if (right.charAt(0)=='$'){
   //do nothing;
       }
       else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
       }
       else if (right.equalsIgnoreCase("-")){
   right=generateMinus(ast);
       }
       else if (right.equalsIgnoreCase("*")){
   right=generateStar(ast);
       }
       else if (right.equalsIgnoreCase("/")){
   right=generateDivide(ast);
       }
       else if (right.equalsIgnoreCase("%")){
   right=generateMod(ast);
       }

       return "("+left+"+"+right+")";
    }

    //method for -
    private String generateMinus(CommonAST s){
CommonAST ast=s;
        String left, right;  //the left of +, and the right of +
        ast=(CommonAST)ast.getFirstChild();
        left= ast.getText();

if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}

        //now the right part of +
       ast=(CommonAST)ast.getNextSibling();
       right=ast.getText();
       if (right.charAt(0)=='$'){
   //do nothing;
       }
       else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
       }
       else if (right.equalsIgnoreCase("-")){
   right=generateMinus(ast);
       }
       else if (right.equalsIgnoreCase("*")){
   right=generateStar(ast);
       }
       else if (right.equalsIgnoreCase("/")){
   right=generateDivide(ast);
       }
       else if (right.equalsIgnoreCase("%")){
   right=generateMod(ast);
       }
```

```
        return "("+left+"-"+right+")";
     }

     //method for *
     private String generateStar(CommonAST s){
CommonAST ast=s;
         String left, right;  //the left of +, and the right of +
         ast=(CommonAST)ast.getFirstChild();
         left= ast.getText();
if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}

         //now the right part of +
        ast=(CommonAST)ast.getNextSibling();
        right=ast.getText();
        if (right.charAt(0)=='$'){
   //do nothing;
        }
        else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
        }
        else if (right.equalsIgnoreCase("-")){
   right=generateMinus(ast);
        }
        else if (right.equalsIgnoreCase("*")){
   right=generateStar(ast);
        }
        else if (right.equalsIgnoreCase("/")){
   right=generateDivide(ast);
        }
```

```
       else if (right.equalsIgnoreCase("%")){
   right=generateMod(ast);
       }

return "("+left+"*"+right+")";

    }

    //method for /
    //a bit more tricky than + - or *, can't divide by 0
    private String generateDivide(CommonAST s){
CommonAST ast=s;
        String left, right;  //the left of +, and the right of +
        ast=(CommonAST)ast.getFirstChild();
        left= ast.getText();

if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}

        //now the right part of +
       ast=(CommonAST)ast.getNextSibling();
       right=ast.getText();

       if (right.charAt(0)=='$'){
   //do nothing;
       }
       else if (right.equalsIgnoreCase("+")){
    right=generatePlus(ast);
       }
       else if (right.equalsIgnoreCase("-")){
```

```
   right=generateMinus(ast);
       }
       else if (right.equalsIgnoreCase("*")){
   right=generateStar(ast);
       }
       else if (right.equalsIgnoreCase("/")){
   right=generateDivide(ast);
       }
       else if (right.equalsIgnoreCase("%")){
   right=generateMod(ast);
       }

       return "("+left+"/"+right+")";

    }

    //method for %
    private String generateMod(CommonAST s){
System.out.println("sleepy head");
CommonAST ast=s;
        String left, right;  //the left of +, and the right of +
        ast=(CommonAST)ast.getFirstChild();
        left= ast.getText();

if (left.charAt(0)=='$'){
    //do nothing;
}
else if (left.equalsIgnoreCase("+")){
    left=generatePlus(ast);
}
else if (left.equalsIgnoreCase("-")){
    left=generateMinus(ast);
}
else if (left.equalsIgnoreCase("*")){
    left=generateStar(ast);
}
else if (left.equalsIgnoreCase("/")){
    left=generateDivide(ast);
}
else if (left.equalsIgnoreCase("%")){
    left=generateMod(ast);
}

        //now the right part of +
       ast=(CommonAST)ast.getNextSibling();
       right=ast.getText();
```

```
        if (right.charAt(0)=='$'){
    //do nothing;
        }
        else if (right.equalsIgnoreCase("+")){
     right=generatePlus(ast);
        }
        else if (right.equalsIgnoreCase("-")){
    right=generateMinus(ast);
        }
        else if (right.equalsIgnoreCase("*")){
    right=generateStar(ast);
        }
        else if (right.equalsIgnoreCase("/")){
    right=generateDivide(ast);
        }
        else if (right.equalsIgnoreCase("%")){
    right=generateMod(ast);
        }


        return "("+left+"%"+right+")";
     }
     /********************End + - + / % *************************
*******************/

     private void generatePerlBlock(CommonAST s){
//simply output the text
CommonAST ast=s;
String code;
code="#perl block code";
codeList.add(code);
code=ast.getText();
code=code.substring(2, code.length()-2);
codeList.add(code);
code="#ends perl block code";
codeList.add(code);
     }

     /*********************end here*************************/


     /**generate code for Echo
      * takes in an ast sub tree
      */
     private void generateEcho(CommonAST s){
         CommonAST ast=s;
        String echoStmt="";
        ast=(CommonAST)ast.getFirstChild();
        String code="";
        if (ast!=null){
            echoStmt=ast.getText();
            //if print a variable
            if (echoStmt.charAt(0)=='$'){
                code="print "+echoStmt +";";
            }
            else if (echoStmt.equalsIgnoreCase("+")){
                code=generatePlus(ast)+"";
                code="print "+code+ ";";
            }
            else if (echoStmt.equalsIgnoreCase("-")){
                code=generateMinus(ast)+"";
                code="print "+code+ ";";
            }
            else if (echoStmt.equalsIgnoreCase("*")){
                code=generateStar(ast)+"";
                code="print "+code+ ";";
            }
            else if (echoStmt.equalsIgnoreCase("/")){
                code=generateDivide(ast)+"";
                code="print "+code+ ";";
            }
            else if (echoStmt.equalsIgnoreCase("%")){
                code=generateMod(ast)+"";
                code="print "+code+ ";";
            }
    else if (echoStmt.equalsIgnoreCase(".")){
code=generateConcat(ast);
code="print "+code+";";
    }
    else if (echoStmt.equalsIgnoreCase("call")){
code=generateCall(ast);
code="print "+code;
    }
            else {
                //print some text, should add double quotes to th
e text
                code="print \""+echoStmt +"\";";
            }

            codeList.add(code);
        }
        else {
            System.out.println("wrong usage of ECHO");
            System.exit(1);
        }
    }

    /**generate code for assign operator "="
     * @param CommonAST s the sub AST tree
     */
    private void generateAssign(CommonAST s){
CommonAST ast=s;
String left; //left has to be some variable
ast=(CommonAST)ast.getFirstChild();

//MARK
left = generateIdentifier(ast);
//left=ast.getText();
//System.out.println("left is "+left);

String code=left;
//right hand side is the tricky part
CommonAST rightAST=(CommonAST)ast.getNextSibling();
String right=rightAST.getText();

//has several possibilities
//first, mathematical operations
if (right.length()==0){
    code=code+" = \"\";";
}
else if (right.equalsIgnoreCase("+")){
    right=generatePlus(rightAST);
    code=code+" = "+right+";";
}
else if (right.equalsIgnoreCase("-")){
    right=generateMinus(rightAST);
    code=code+" = "+right+";";
}
else if (right.equalsIgnoreCase("*")){
    right=generateStar(rightAST);
    code=code+" = "+right+";";
}
else if (right.equalsIgnoreCase("/")){
    right=generateDivide(rightAST);
    code=code+" = "+right+";";
}
else if (right.equalsIgnoreCase("%")){
    right=generateMod(rightAST);
    code=code+" = "+right+";";
}
//an array
else if (right.equalsIgnoreCase("{")){
    code=left.replace('$', '@'); //@ is the symbol for an
 arry in perl
    int childNumber=rightAST.getNumberOfChildren();
    CommonAST list=(CommonAST)rightAST.getFirstChild();
    if (!(list.getText()).equalsIgnoreCase("}")){
right=" ("+list.getText();
for (int i=1; i<childNumber; i++){
    list=(CommonAST)list.getNextSibling();
    right=right+", "+list.getText();
}
right=right+")";
code=code+" = "+right+";";
    }
    else {
//empty array
right="()";
code=code+" = "+right+";";
    }
}
//regular expression
else if (right.equalsIgnoreCase("@")){
    right=generateAt(rightAST);
    code=left.replace('$','@');
    code=code+" = "+right+";";
}
else if (right.equalsIgnoreCase(".")){
    right=generateConcat(rightAST);
    code=code+" = "+right+";";
}
else if (right.equalsIgnoreCase("call")){
    right=generateCall(rightAST);
    code=code+" = "+right;
}
else if (right.charAt(0)<='9'&&right.charAt(0)>='0'){
    code = code + " = " +right+ ";";
}
else if (right.charAt(0)!='$'){
    right="\""+right+"\"";
    code=code+" =" +right+";";
}
else if (right.length()==0){
    code=code+" = \";";
```

```
}
else{
    //MARK
    right=generateIdentifier(rightAST);
    code = code + " = " + right + ";";

    //right=rightAST.getText();
    //code=code+" = "+right+";";
}


codeList.add(code);
//an function??? not implemented yet and others????

    }

    /** generate code for httpget operator
     * @param the sub AST tree
     */
    private void generateHTTPGET(CommonAST s){
        //should have only two child for := operator
        String addrHolder="";
        CommonAST ast=s;
        if (s.getFirstChild()!=null){
            ast=(CommonAST)ast.getFirstChild();
            addrHolder=ast.getText();
            /* use the variable name as the temperatory html file
 name
            * have to get rid of the leading $ sign */
        }
else {
            System.out.println("Wrong usage of HTTPGET operator :
=");
            System.exit(1);  //exit here??
        }


String code="";
        addrHolder=addrHolder.substring(1);  //get rid of the lea
ding $
ast=(CommonAST)ast.getNextSibling();
        String addr=ast.getText();
if (addr.equalsIgnoreCase(".")){
    addr=generateConcat(ast);
    code+="open(fileOUT, \">tmp.txt\") or dienice (\"Can'
t open tmp.txt for writing\");";
```

```
    codeList.add(code);
    code="print fileOUT "+addr+" ;\n";
    codeList.add(code);
    code="close(fileOUT);";
    codeList.add(code);
    code="system('wget -q -O "+addrHolder+".txt -i tmp.tx
t');";
    codeList.add(code);
    code="system('rm tmp.txt');";
    codeList.add(code);
}
else if (!addr.equalsIgnoreCase("")&&addr.charAt(0)=='$')
{
    //write the value of the variable into a file, and do
 wget file
    code+="open(fileOUT, \">tmp.txt\") or dienice (\"Can'
t open tmp.txt for writing\");";
    codeList.add(code);
    code="print fileOUT "+addr+" ;\n";
    codeList.add(code);
    code="close(fileOUT);";
    codeList.add(code);
    code="system('wget -q -O "+addrHolder+".txt -i tmp.tx
t');";
    codeList.add(code);
    code="system('rm tmp.txt');";
    codeList.add(code);
}

else {
            //always have "" surround the http address, so get ri
d of them
    //update, lex/parser will take the quotes off
    addr=addr.substring(0, addr.length());

    //download the html file from the address, and save t
he file in addrHolder.txt

    code="system('wget -q -O - "+addr+"  > " +addrHolder
+ ".txt');";
    //add the line of code into the linkedlist
    codeList.add(code);
}
//open the file, and put everything in an array
code="open INFILE, \""+addrHolder+".txt\";";
codeList.add(code);
```

```
code="@"+addrHolder+"=<INFILE>;";
codeList.add(code);
code="close INFILE;";
codeList.add(code);


//MARK
//join into a single line
// code = "@" + addrHolder + "=join('',@" + addrHold
er+ ");";
//codeList.add(code);

//rm the tmp file
code="system ('rm "+addrHolder+".txt');";
codeList.add(code);
        String httpVar='$'+addrHolder;
        httpVariableList.add(httpVar);
code=httpVar+" = \""+addr+"\";";
codeList.add(code);
//should be working fine now....hopefully!
    }


    // generateIdentifier, main() written by T. Mark Kuba
    //
    // to generate identifiers if they are arrays
    private String generateIdentifier(CommonAST s) {

String code = "";
String id = s.getText();

//stem.out.println("Now in generateIdentifier");

//System.out.println(id);

// check to see if array

if (s.getNumberOfChildren() != 0) {

    id = id.replace('$','@');

    //System.out.println("id is now " + id);

    code = code + id +  "[";
    //System.out.println("here it is!" + v.getText());

    // Deal with the inner bracket stuff (i.e. $a[$x+5] =
 3;)

    AST t = s.getFirstChild();
    CommonAST rightAST = (CommonAST)t.getNextSibling();

    String right=rightAST.getText();
    //has several possibilities
    //first, mathematical operations
    if (right.equalsIgnoreCase("+")){
right=generatePlus(rightAST);
code+=right;
    }
    else if (right.equalsIgnoreCase("-")){
right=generateMinus(rightAST);
code=code+right;
    }
    else if (right.equalsIgnoreCase("*")){
right=generateStar(rightAST);
code=code+right;
    }
    else if (right.equalsIgnoreCase("/")){
right=generateDivide(rightAST);
code=code+right;
    }
    else if (right.equalsIgnoreCase("%")){
right=generateMod(rightAST);
code=code+right;
    }
    else if (right.charAt(0) < '9' && right.charAt(0) > '
0') {
code = code + right;
    }
    else if (right.charAt(0)!='$'){
right="\""+right+"\"";
code=code+right;
    }
    else{
//MARK
right=generateIdentifier(rightAST);
code = code + right;

//right=rightAST.getText();
//code=code+" = "+right+";";
```

```
    }



    //code = code + v.getText();


    code = code + "]";
} else {

    code = id;
}


return code;
    }


    public static void main(String args[]) throws Exception{

try {
    if (args.length < 1) {
System.out.println("Usage: java Main <yolt file>"
);
System.exit(0);
    }
    File f = new File(args[0]);
    //L lexer = new L(new DataInputStream(System.in));
    L lexer = new L(new FileInputStream(f));
            P parser = new P(lexer);


    parser.program();


    AST c = parser.getAST();
    System.out.println("Here's the tree!");
    System.out.println(c.toStringTree());

    System.out.println("\n");

    CodeGenerator g = new CodeGenerator((CommonAST)c);
    g.generate(g.getTree());

    g.result();
```

```
        } catch(Exception e) {
    e.printStackTrace();
        }

    }

}
```

11.4 Bash Scripts, Misc.

11.4.1 make_ok

```
#!/bin/bash

# By; Lukas Dudkowski
# folds files to a specified width, makes safe for LaTeX, or printing


tempfile="temp"

if [ $# -eq 2 ]; then
fold $1 --width=$2 > $tempfile
more $tempfile > $1
else
echo "wrong usage"
fi

rm $tempfile
```

11.4.2 test_diff

```
#!/bin/bash

# @author: Lukas Dudkowski
# script used to generate test logs, regression testing
# generates proper TeX files used for final report


globallog=yolt_test.log
latex_log=latex.log

output_tex=lexer_test.tex

rm -f $output_tex
rm -f $globallog
rm -f $latex_log

error=0

Check() {
echo "Trying file: $1"
basename=$(echo $1 | sed 's/.y$//')
reffile=$(echo $1 | sed 's/.y$/.REFERENCE/')

lexeroutputfile=${basename}.lex_tree   #where we redirect output
 of lexer
difffile=${basename}.diff  #see if they differ
echo -n "Parsing test on: $basename..."
echo "Parsing $1" 1>&2
#run java Main <yolt file> and output to
java Main $1 > $lexeroutputfile 2>&1 || {
echo "FAILED: yolt lexer/parser terminated"
echo "FAILED: yolt lexer/parser terminated" 1>&2
error=1; return 1
}
diff -b $reffile $lexeroutputfile > $difffile 2>&1 || {
echo "FAILED: output mismatch"
echo "FAILED: output mismatch" 1>&2
error=1 ; return 1
}

#rm $lexeroutputfile $difffile
echo "file PASSED"
echo "file PASSED" 1>&2
}
```

```
Generate_latex() {


basename=$(echo $1 | sed 's/.y$//')
reffile=$(echo $1 | sed 's/.y$/.REFERENCE/')
latex_gen_file=${basename}.lex_tree
difffile=${basename}.diff

echo "YOLT source file: $1" >> $output_tex

cat newline.file $1 newline.file spacer_lex.file newline.file $la
tex_gen_file newline.file spacer_ref.file newline.file $reffile n
ewline.file spacer.file newline.file >> $output_tex

echo "Diff with REFERENCE:" >> $output_tex
cat $difffile newline.file spacer.file newline.file >>$output_tex

}

for file in test_*.y
do
Check $file 2>> $globallog
done

#make the latex output head...
cat head.file verbatim.file >> $output_tex

for file in test_*.y
do
        echo "Trying to cat: $file"
Generate_latex $file
done

#make the latex output tail

echo "DIFF RESULTS FOR THE TEST RUN:" >> $output_tex
cat $globallog newline.file >> $output_tex

echo "END OF TEST" >> $output_tex
cat tail_verbatim.file tail.file >> $output_tex

#do some cleanup
rm *.diff
rm *.lex_tree
```

```
exit $error
```

11.4.3 yolt

```
#!/bin/bash

# @author: Lukas Dudkowski
# runs the actual yolt stuff, takes input from command line on ou
tput file
# can delete intermediate perl...   but left out for debugging.


if [ $# -lt 1 ]; then
echo "Usage <yolt> input_file [output_file]"
exit 1
fi
if [ $# -eq 1 ]; then
basename=$(echo $1 | sed 's/.y$//')
perl_file=${basename}.pl
java Main $1 > $perl_file
perl $perl_file
exit 0
fi

if [ $# -eq 2 ]; then
     basename=$(echo $1 | sed 's/.y$//')
perl_file=${basename}.pl
java Main $1 > $perl_file
perl $perl_file > $2
exit 0
else
        echo "Usage <yolt> input_file [output_file]"
exit 1
fi

#cleanup ?
#rm $perl_file
```

11.5 YOLT information scraping examples

11.5.1 edwards.y

```
# Principal Author: T. Mark Kuba
#
# download "crazy" photos from Stephen Edwards's web site
#

begin

$home = "http://www1.cs.columbia.edu/~sedwards/";
$page := "http://www1.cs.columbia.edu/~sedwards/personal.html";

$tags = "<a href=\"(.*crazy.*)index\.html\">";

$a1 = $tags @ $page;

foreach $link in $a1 {

  echo "<p>" .$link . "<br>\n\n\n";

  $picpage := $home . $link;

  $tags2 = "<img src=\"(.*)\" alt=\".*\" width=\".*\" height=\".*
\">";

  $pics = $tags2 @ $picpage;

  foreach $pic in $pics {
    echo "<img src=\"" . $home.$link . $pic . "\">\n";
  }

}

end
```

11.5.2 webcomics.y

```
### Principal Author: T. Mark Kuba
#
# download web comics
#

### web comics script ###
begin

declare scrape($title,$url,$tags);

### achewood

$tags = "<img src=\"(/dat/comic/.*)\" title=\".*\" hspace=\"10\"
vspace=\"0\" border=\"0\">";
call scrape("Achewood","http://www.achewood.com",$tags);

### superosity

$tags = "<img ALT=\".*\" BORDER=0 SRC=\"(/comics/sup.*)\" HEIGHT=
\".*\" WIDTH=\".*\">";
call scrape("Superosity","http://www.superosity.com",$tags);

### doonesbury

$doonesbury := "http://www.doonesbury.com/strip/dailydose/";
$tags = "<img src=\"(.*images.ucomics.com/comics/db/.*)\" border=
\"0\" title=\".*\">";

$ar = $tags @ $doonesbury;

echo "<a href=\"" . $doonesbury . "\">Doonesbury</a><br>\n";
foreach $el in $ar {
  echo "<img src=\"" . $el . "\"><p>";
}

### dilbert

$tags = "<img src=\"(/comics/dilbert/archive/images/dilbert200.*)
\">";
call scrape("Dilbert","http://www.dilbert.com",$tags);

### sinfest

$tags = "<img ALT=\".*\" BORDER=0 SRC=\"(/comics/sf.*)\" HEIGHT=\
".*\" WIDTH=\".*\">";
call scrape("Sinfest","http://www.sinfest.net",$tags);

### real life comics

$tags = "<img src='(.*)' border=0>";
call scrape("Real Life Comics","http://www.reallifecomics.com",$t
ags);

### boxjam

$tags = "<img ALT=\".*\" BORDER=0 SRC=\"(/comics/bj.*)\" HEIGHT=\
".*\" WIDTH=\".*\">";
call scrape("Boxjam's Doodle","http://www.boxjamsdoodle.com",$tag
s);

### goats

$tags = "<img src=\"(/comix.*)\">";
call scrape("Goats","http://www.goats.com",$tags);

### rpg world

$tags = "<img ALT=\".*\" BORDER=0 SRC=\"(/comics/.*)\" HEIGHT=\".
*\" WIDTH=\".*\">";
call scrape("RPG World","http://www.rpgworldcomic.com",$tags);

### pvp

$tags = "<img SRC=\"(archive/.*)\">";
call scrape("PVP","http://www.pvponline.com/",$tags);

### penny arcade

$home := "http://www.penny-arcade.com/";
$page := "http://www.penny-arcade.com/view.php3";

$tags = "<img src=\"(images/200.*)\" ALT=\"\">";

$ar = $tags @ $page;

echo "<a href=\"" . $home . "\">Penny Arcade</a><br>\n";
foreach $el in $ar {
  echo "<img src=\"". $home . $el . "\"><p>\n\n";
}
```

```
### you damn kid

$tags = "<img ALT=\".*\" BORDER=0 SRC=\"(/comics/ydk.*)\" HEIGHT=
\".*\" WIDTH=\".*\">";
call scrape("You Damn Kid","http://www.youdamnkid.com",$tags);




### functions to scrape web pages

function scrape($title,$url,$tags) {
  $page := $url;

  $ar = $tags @ $page;

  echo "<a href=\"" . $url . "\">" . $title . "</a><br>\n";
  foreach $img in $ar {
    echo "<img src=\"" . $page . $img . "\"><p>\n\n";
  }

}


end
```

11.5.3 daily.y

```
# Principal Author: T. Mark Kuba
#
# scrape daily news and headlines
#

begin

echo "Here are your daily news!<p>";

echo "<p>";

#big table
echo "<TABLE BORDER=0><TR><TD VALIGN=\"TOP\">";

##################
# slashdot

$slashdot := "http://slashdot.org";

$tags = "<a href=\"(.*)\" TITLE=\"\"><b>Read More...</b></a>";

$story_headline = "FACE=\"arial,helvetica\" SIZE=\"4\" COLOR=\"#F
FFFFF\"><b>(.*)</b></font></td>";

$array = $tags @ $slashdot;
$array2 = $story_headline @ $slashdot;

$x = 0;

echo "<table border=0>";
echo "<tr><td bgcolor=\"#008080\"><font color=\"#FFFFFF\"><strong
>Slashdot.org</strong></font></td></tr>";
echo "<tr><td bgcolor=\"#CCCCCC\">";

foreach $element in $array {

  echo "  " .$array2[$x];
  echo "  [<a href=\"". "http:" . $element . "\">link</a>]<br>";
  $x = $x+1;
}

echo "</td></tr></table>";
echo "<p>";

##################
# ars technica


$ars := "http://www.arstechnica.com";

$tags = "<h2>(.*)</h2>";

$array = $tags @ $ars;

echo "<table border=0><tr><td bgcolor=\"#000080\"><font color=\"#
FFFFFF\"><strong>Ars Technica</strong></font></td></tr>";

echo "<tr><td bgcolor=\"#000000\">";
foreach $element in $array {

  echo $element . "<br>";

}
echo "</td></tr></table>";

echo "</TD><TD VALIGN=\"TOP\">";

##################
# cnn search

$cnn := "http://www.cnn.com";

$search_term = "SARS";

$story_tags = "<a href=\"(/2003/.*)\">.*" . $search_term. ".*</a>
";

$story_tags2 = "<a href=\"/2003/.*\">(.*" .$search_term .".*)</a>
";

$myArray = $story_tags @ $cnn;
$myArray2 = $story_tags2 @ $cnn;


$x = 0;

echo "<table border=0><tr><td bgcolor=\"#FF0000\"><font color=\"#
FFFFFF\"><strong>CNN search for \"" . $search_term . "\"</strong>
```

```
</font></td></tr>";

echo "<tr><td bgcolor=\"#CCCCCC\">";
foreach $el in $myArray {

  echo "<a href=\"" . $cnn . $el . "\">" . $myArray2[$x]  ."</a><
br>";
  $x = $x+1;

}
echo "</td></tr></table>";

echo "<p>";

################
# aint it cool news


$aicn := "http://www.aintitcoolnews.com/";

$search_term = "Matrix";

$tags = "<a href=\"(display\.cgi.*)\">.*" . $search_term . ".*</a
>";
$tags2 = "<a href=\"display\.cgi.*\">(.*" . $search_term . ".*)</
a>";

$a1 = $tags @ $aicn;
$a2 = $tags2 @ $aicn;


echo "<table border=0><tr><td bgcolor=\"#996600\"><font color=\"#
FFFFFF\"><strong>Ain't It Cool News search for " . $search_term .
"</strong></font></td></tr>";

echo "<tr><td bgcolor=\"#CCCCCC\">";

$x = 0;
foreach $el in $a1 {

  echo "<a href=\"" . $aicn . $el . "\">" . $a2[$x] . "</a><br>";
  $x = $x + 1;

}
echo "</td></tr></table>";
echo "<p>";

################
# kuro5hin


$k5 := "http://www.kuro5hin.org";

$tags = "<font FACE=\"verdana, arial, helvetica, sans-serif\"><b>
<a HREF=\"(/story/.*)\" style=\"text-decoration:none;\"><font col
or=\"#000000\">.*</font>";

$tags2 = "<font FACE=\"verdana, arial, helvetica, sans-serif\"><b
><a HREF=\"/story/.*\" style=\"text-decoration:none;\"><font colo
r=\"#000000\">(.*)</font>";


$ar1 = $tags @ $k5;
$ar2 = $tags2 @ $k5;


echo "<table border=0><tr><Td bgcolor=\"#006699\"><font color=\"#
FFFFFF\"><strong>Kuro5hin</strong></font></td></tr>";
echo "<tr><td bgcolor=\"#CCCCCC\">";
$x = 0;
foreach $el in $ar1 {

  echo "<a href=\"" . $k5 . $el . "\">" . $ar2[$x] . "</a><br>\n"
;
  $x = $x+1;
}
echo "</td></tr></table>";

echo "<p>";

################
# The Onion


$onion := "http://www.theonion.com";

$tags1 = "<a href=\"(/onion3917/.*)\" CLASS=\"headline\"><b>.*</b
>";

$tags2 = "<a href=\"/onion3917/.*\" CLASS=\"headline\"><b>(.*)</b
>";
```

```
$a1 = $tags1 @ $onion;
$a2 = $tags2 @ $onion;

$x = 0;

echo "<table border=0><tr><Td bgcolor=\"#007733\"><font color=\"#
FFFFFF\"><strong>The Onion</strong></font></td></tr>";
echo "<tr><td bgcolor=\"#CCCCCC\">";
foreach $el in $a1 {
  echo "<a href=\"" . $onion . $el . "\">" . $a2[$x] ."</a><br>\n
";
  $x = $x + 1;
}
echo "</td></tr></table>";

echo "</TD></TR></TABLE>";

echo "<p>";



################
# web comics
$goats := "http://www.goats.com";
$tags = "<img src=\"(/comix.*)\">";

$array = $tags @ $goats;

echo "<h5><a href=\"" .$goats . "\">goats</a></h5>";
foreach $element in $array {
  echo "<img src=\"" . $goats . $element . "\">";
}

echo "<p>";

$dilbert := "http://www.dilbert.com";
$tags = "<img src=\"(/comics/dilbert/archive/images/dilbert200.*)
\">";

$array = $tags @ $dilbert;

echo "<h5><a href=\"" .$dilbert . "\">Dilbert</a></h5>";
foreach $element in $array {
  echo "<img src=\"" . $dilbert . $element . "\">";
}
```

```
end
```

11.5.4 tooth.y

```
# @author: Lukas Dudkowski
# get stuff from toothpastefordinner
# used for integration testing, shown during presentation

begin

$toothpaste_home="http://www.toothpastefordinner.com/";
$toothpaste:="http://www.toothpastefordinner.com/archives-sum02.p
hp";

$tags="<a href=\"(.*)\">.*hamster.*</a>";

$elements = $tags @ $toothpaste;

foreach $x in $elements {

  echo "<img src=\"".$toothpaste_home.$x."\""."><br>";
  echo "\n";

}

end
```

11.5.5 yahoo.y

```
# @author: Lukas Dudkowski
# scrape yahoo.com for any links...
# display info, used for tutorial


begin

$page:="http://www.yahoo.com";

$tags = "<a href=.*>(.*)</a>";

$links = $tags @ $page;

foreach $line in $links{

echo $line;
echo "\n";
}

end
```