gscc

A General Search and Compare Compiler

gscc is a text manipulation language that rivals existing programmatic solutions. It is compact, intuitive and lightweight, giving programmers a means to quickly manipulate their text-based targets.

Eric [G]arrido
Russel [S]antillanes
Casey [C]allendrello
Ho Yin [C]heng

gscc Language Overview

- Text manipulation
  - Much like AWK
  - Regular Expressions
  - Simple commands
    - set, replace, delete, insert, print/prerr, and more
  - Feature: Location Variables
    - @match, @line

High Level Overview

text input

```
Mary had a little lamb
With fur as white as snow.
```

regex block

```
[wh.*sn..] line {
set @match, “blue as water”;
}
```

@match = “white as snow”

regex block

```
[as] global {
set @match, “comme”;
print @line;
}
```

High Level Overview

text input

```
Mary had a little lamb
With fur as white as snow.
```

regex block

```
[wh.*sn..] line {
set @match, “blue as water”;
}
```

regex block

```
[as] global {
set @match, “comme”;
print @line;
}
```

@match = “as”;

High Level Overview

text input

Mary had a little lamb
With fur as white as snow.

regex block

```
[wh.*sn..] line {
set @match, “blue as water”;
}
```

regex block

```
[as] global {
set @match, “comme”;
print @line;
}
```

@match = “as”;

High Level Overview

text input

```
Mary had a little lamb
With fur as white as snow.
```

regex block

```
[wh.*sn..] line {
set @match, “blue as water”;
}
```

@match = “white as snow”

regex block

```
[as] global {
set @match, “comme”;
print @line;
}
```

@match = “as”;

With fur comme blue as water.
With fur comme blue comme water.

Architecture and Implementation

Architecture and Implementation

Basics

- Front end: Lexer, Parser
- Back end: walker, interpreter
  - Type system
  - Initial setup: Walker detects program structure, Interpreter remembers AST nodes and walks, later, as needed.

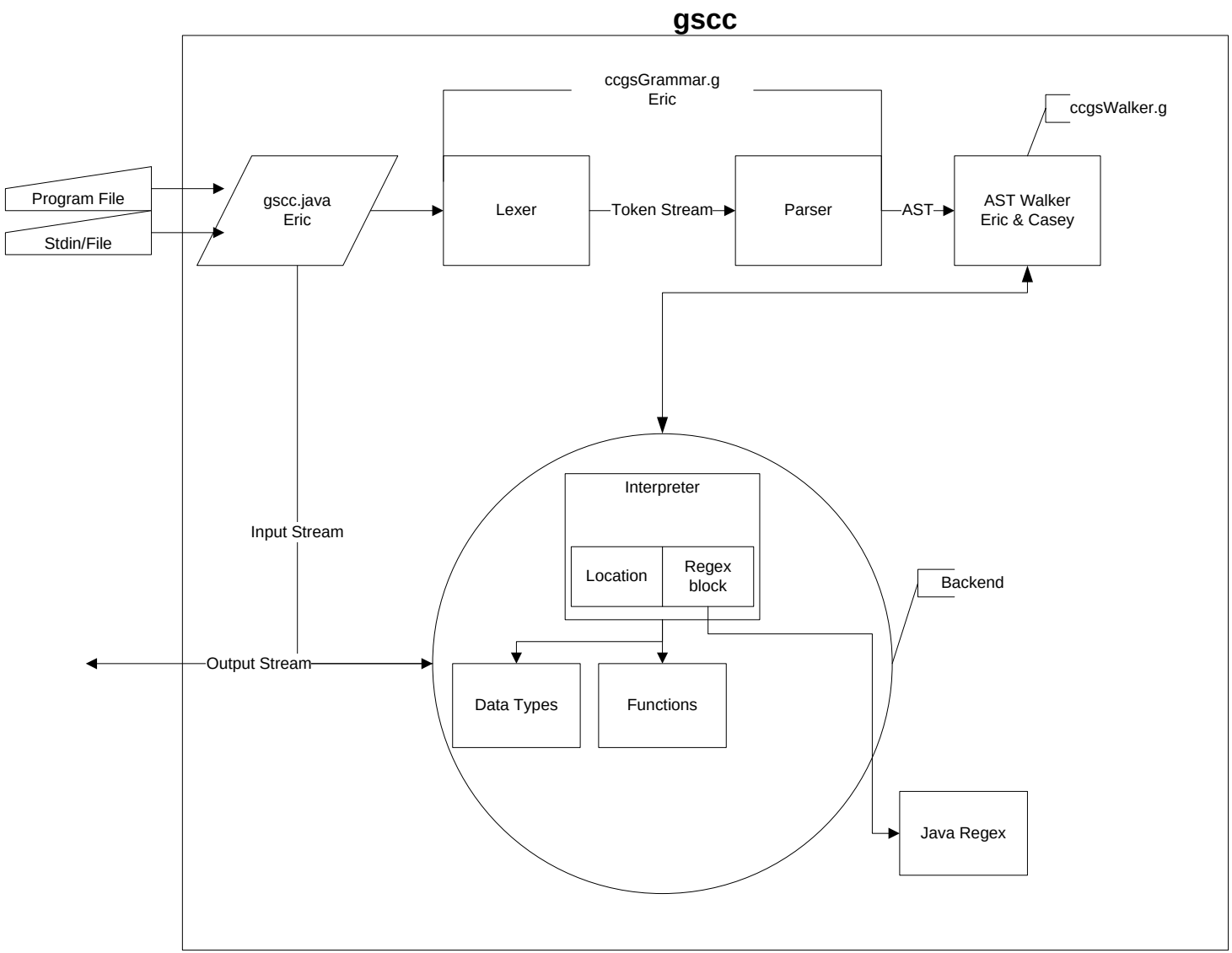
Architecture and Implementation

Interface: Interpreter.java

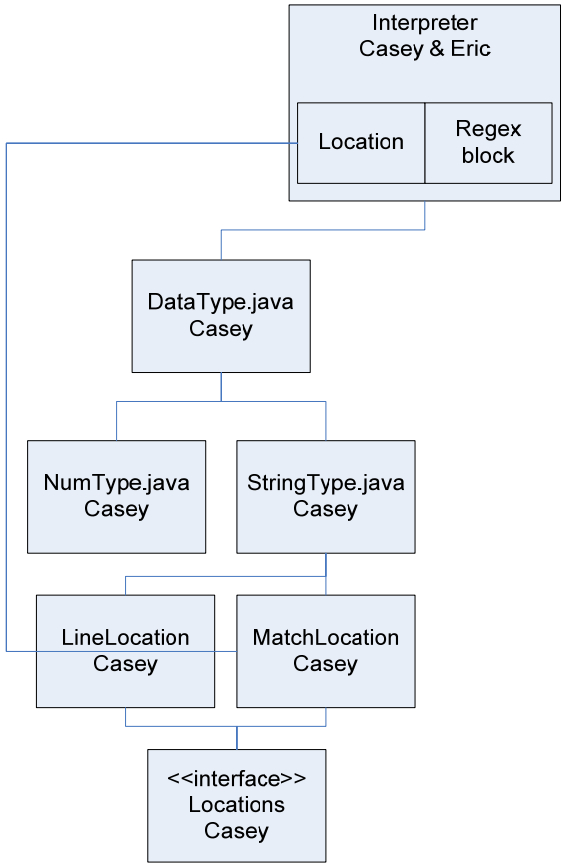
- Interacts with walker to execute program

```
public interface Interpreter {

    public void registerFunction(String name, ParamList paramlist, AST node)

    public DataType callFunction(String name, ExpressionList explist)

    public void runCommand(String name, String target, ExpressionList exprlist)

    public DataType getVariable(String name);
    public DataType getAttrib(String name, String attrName);

    public void registerRegexBlock(String regex, String type, AST node);

    public void runInput(java.io.BufferedReader in, AST program);

    public void setReturn(DataType value);


    //plus flow-control
}
```

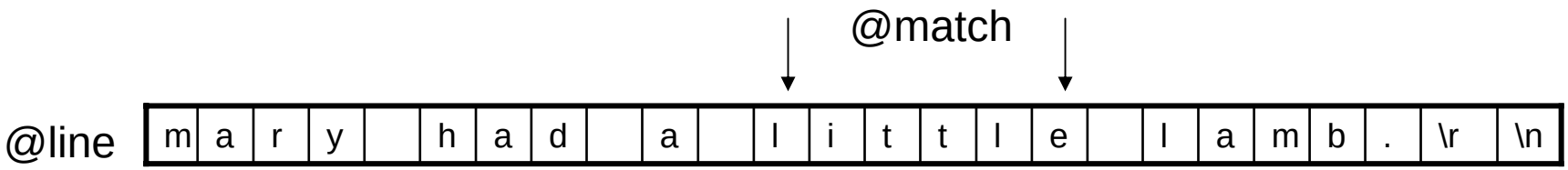
Architecture



Type Hierarchy



Locations



- Represented as a linked list internally
- changing @match automatically changes @line
- changing @line may change @match
  - the *replace* @line command may overwrite @match
  - @match can become undefined

Tutorial

gscc basics

- All statements must be within regex blocks and function definitions with the exception of the SET command.

- Statement can be a command or a function call.

Your first program

```
[H*] line {
   print $foo() + “\n”;
}
func $foo(){
   return “Hello World”;
}
```

Making it more useful

- Locations give you access to the incoming text
  - @line, @match are global variables.
  - @match is the text that matches a regular expression
  - @line is the whole line being operated on
- Modifications to locations affect the next regular expresson block

Finding 404s

- Example: Parsing an apache logfile
  - Say you want to find words that are misspelled resulting in a 404

Apache logfile format:

221.116.200.62 - - [19/Dec/2005:17:08:36 -0500] "POST /xmlsrv/xmlrpc.php HTTP/1.1" 404 278

A simple example

```
[".*”\s404] line {
  print $substr(@match, 0, @match.length-
  4) + “\n”;
}
```

Refining this

- Somebody is probing for vulnerabilities. You want to ignore this specific access

```
[xmlrpc\.php] line { set @line, “”;}
[".*”\s404] line {
  print $substr(@match, 0, @match.length-4) +
  “\n”;
}
```

A More Complete Program

- Now say we want to count the number of 404's as well as print them out.

```
set $count, 0;
[xmlrpc\.php] line { set @line, “”;}
[".*”\s404] line {
   set $count, $count+1;
   print $count + “\t”;
   print $substr(@match, 0, @match.length-4) +
   “\n”;
}
```

Other Commands

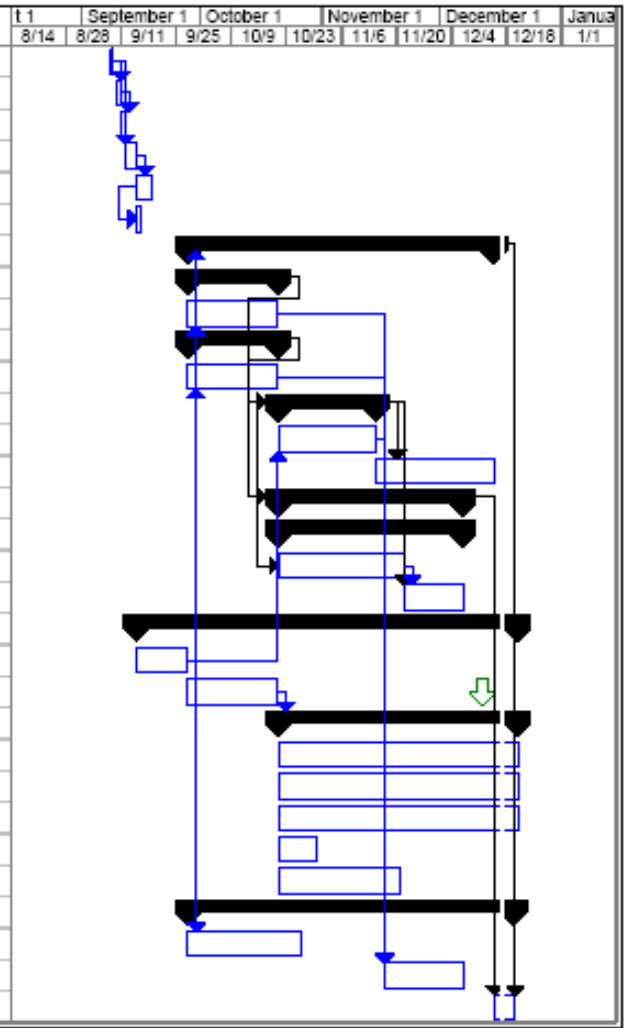
- The previous example used only a small set of the available commands.
- Other commands include:

  replace, delete, insert, prerr
- We also have location attributes and the built in function #length for use.

Summary

Project Plan

ID	Task Name	Ownership	Duration	Start	Finish	Predecessors
1	Choose Teams	All	1 day	Thu 9/8/05	Thu 9/8/05	
2	Select a Meeting Time	All	1 day	Sat 9/10/05	Sat 9/10/05	1
3	Pick A Team Leader	All	1 day?	Sun 9/11/05	Sun 9/11/05	2
4	Install Eclipse	All	3 days	Mon 9/12/05	Wed 9/14/05	1
5	Set up antireclipse	All	3 days	Thu 9/15/05	Sun 9/18/05	4
6	Create a SVN Repository	Casey	1 day	Thu 9/15/05	Thu 9/15/05	5SS
7	**Front End**	**Eric**	**56 days?**	**Wed 9/28/05**	**Wed 12/14/05**	**20**
8	**Lexer**	**Casey, Eric**	**17 days?**	**Wed 9/28/05**	**Thu 10/20/05**	
9	Grammar		17 days?	Wed 9/28/05	Thu 10/20/05	20
10	**Parser**	**Casey, Eric**	**17 days?**	**Wed 9/28/05**	**Thu 10/20/05**	
11	Grammar		17 days?	Wed 9/28/05	Thu 10/20/05	20
12	**Tree Walker**	**Eric**	**17 days?**	**Fri 10/21/05**	**Mon 11/14/05**	**8,10**
13	Grammar		17 days?	Fri 10/21/05	Mon 11/14/05	20
14	Main Interface		22 days?	Tue 11/15/05	Wed 12/14/05	12
15	**Backend**	**Casey**	**33 days?**	**Fri 10/21/05**	**Tue 12/6/05**	**10,8**
16	**Interpreter**		**33 days?**	**Fri 10/21/05**	**Tue 12/6/05**	
17	Data Types		22 days?	Fri 10/21/05	Mon 11/21/05	12SS
18	Functions		11 days?	Tue 11/22/05	Tue 12/6/05	17,12
19	**Documentation**	**Russel**	**70 days?**	**Thu 9/15/05**	**Tue 12/20/05**	
20	Project Proposal	All	10 days?	Thu 9/15/05	Tue 9/27/05	
21	Initial LRM	Russel, Mike	17 days?	Wed 9/28/05	Thu 10/20/05	
22	**Final Report**	**Russel**	**43 days?**	**Fri 10/21/05**	**Tue 12/20/05**	**21**
23	Introduction		43 days?	Fri 10/21/05	Tue 12/20/05	
24	Language Tutorial		43 days?	Fri 10/21/05	Tue 12/20/05	
25	LRM		43 days?	Fri 10/21/05	Tue 12/20/05	
26	Project Plan		6 days	Fri 10/21/05	Sun 10/30/05	
27	Architectual Design		21 days	Fri 10/21/05	Sun 11/20/05	
28	**Testing**	**Mike**	**59 days?**	**Wed 9/28/05**	**Mon 12/19/05**	
29	Test Plan		21 days	Wed 9/28/05	Wed 10/26/05	20
30	Test Cases	Mike, Eric	14 days?	Thu 11/17/05	Tue 12/6/05	9,11,13
31	Regression Testing	All?	3 days?	Thu 12/15/05	Mon 12/19/05	15,7



Project: PLT
Date: Sat 12/17/05

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

Page 1

Lessons Learned

- Start early, Start early, Start early. There is no better feeling in the world than finishing your duties or a project ahead of schedule. There is no worse feeling than missing a hard deadline.

- Deadlines are an important thing to both know and create. Knowing when what is due keeps people on track and will prevent any unforeseen mishaps. They can also serve as a way to enforce team members to submit work if needed.

More Lessons

- Never compromise on your environment. Spending a few hours setting it up in the beginning is easily the best thing you can do with your time.

- Constant communication beyond team meetings can help to keep things flowing. If any of the members isn't performing for whatever reason, having people there to remind them serves as a good motivating factor.

- If you don't know the answer chances are someone else in your group will or will at the least be able to point you in the right direction. Keep asking until you get the answer you want.

Essentials

- http://www.eclipse.org -- Eclipse IDE
- http://ANTLReclipse.sourceforge.net/ -- ANTLR plugin for eclipse
- http://subversion.tigris.org/ -- Subversion version control system
- http://subclipse.tigris.org/ -- Eclipse SVN plugin
- http://e-p-i-c.sourceforge.net/ -- Eclipse PERL plugin
- http://www.apple.com/macosx/ -- The best development platform there is