

# Phoenix-Based Clone Detection using Suffix Trees

Robert Tairas

<http://www.cis.uab.edu/tairasr/clones>

Advisor: Dr. Jeff Gray



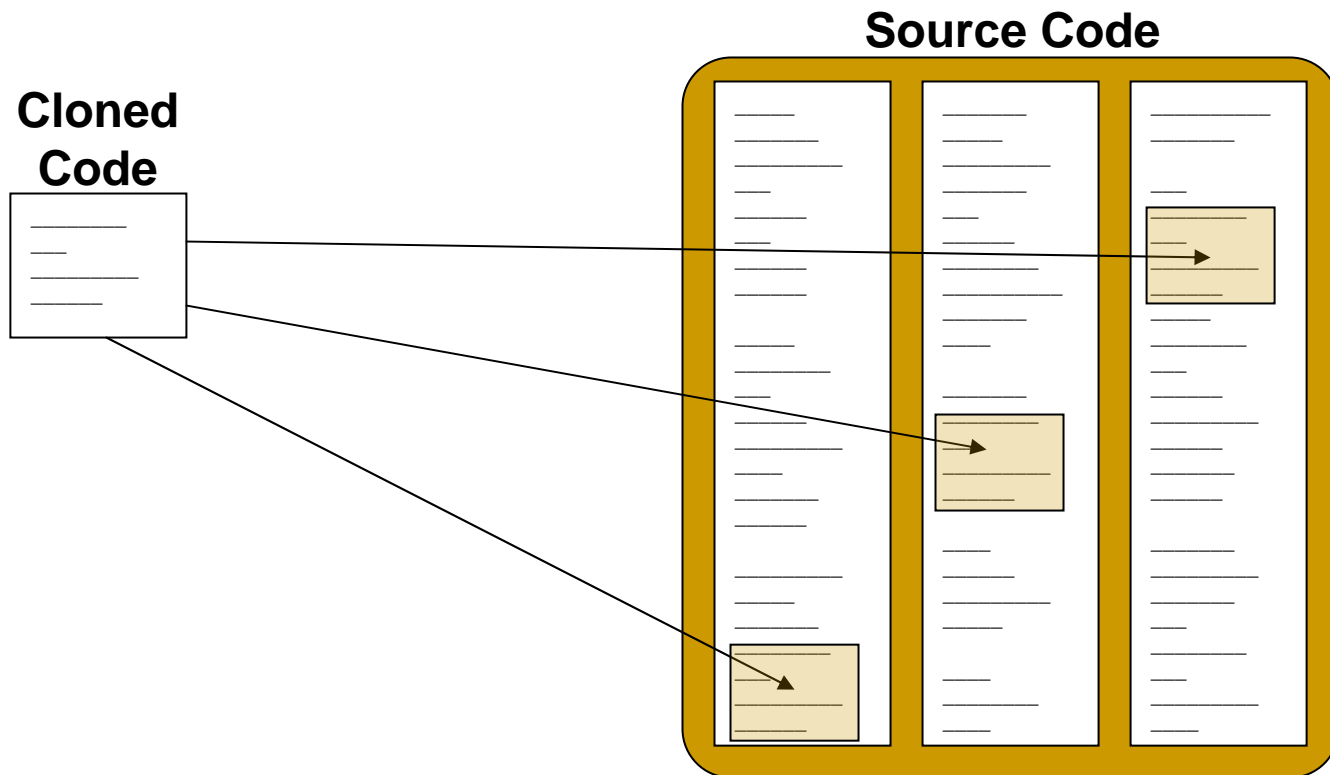
Department of Computer and Information Sciences

University of Alabama at Birmingham

ACM Southeast Conference  
*Melbourne, FL*  
March 11, 2006

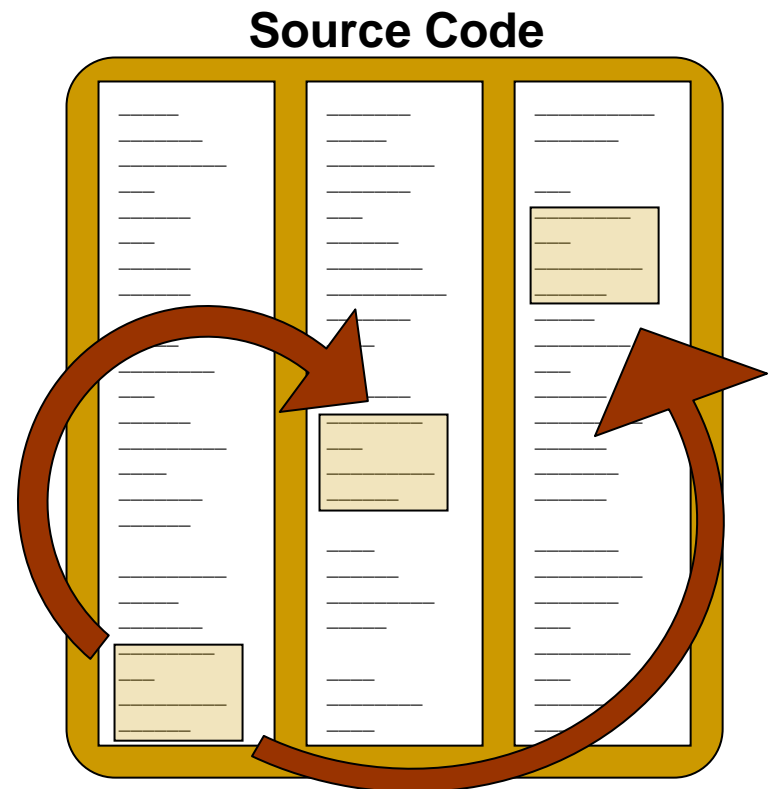
# Code Clones

- A sequence of statements that are duplicated in multiple locations in a program



# Clones in Source Code

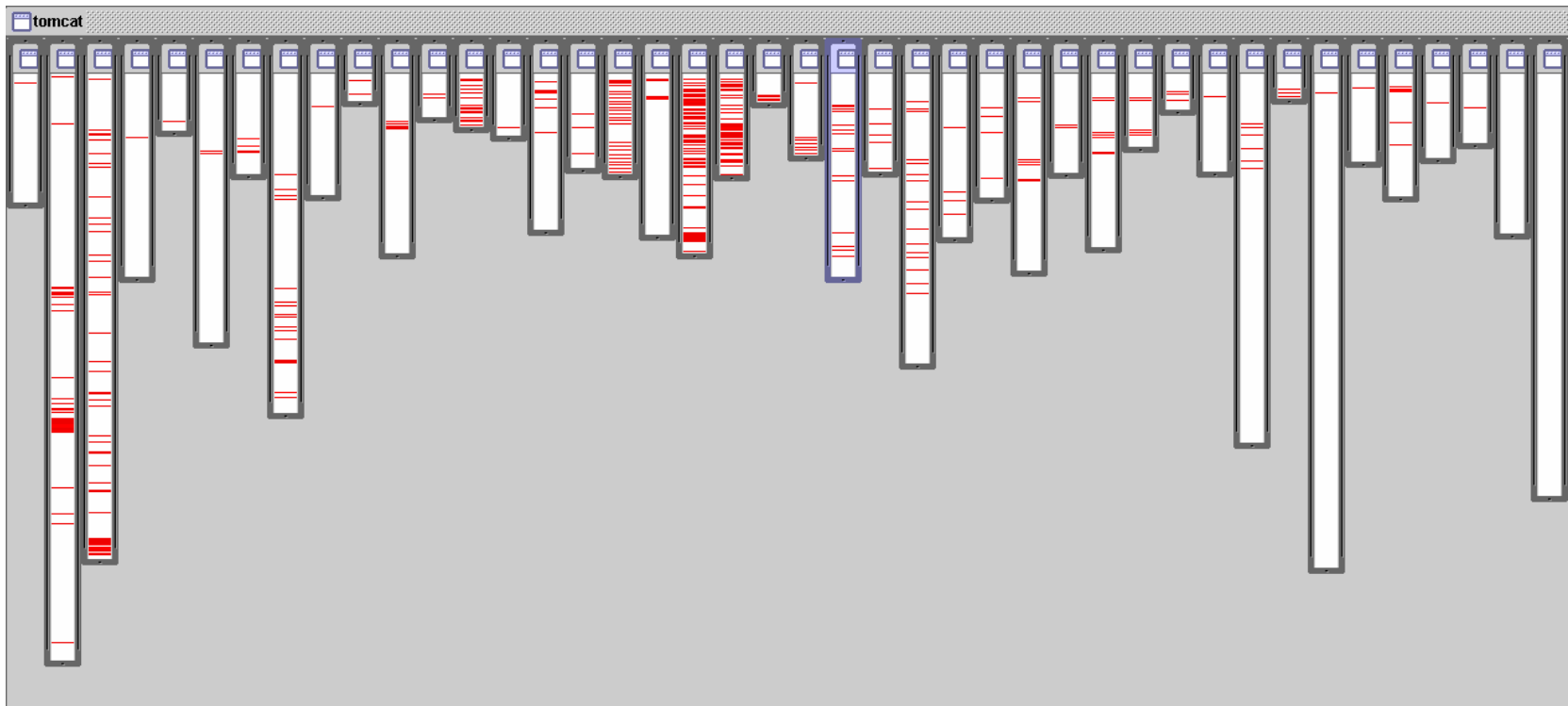
- Copy-and-paste parts of code from one location to another
  - The copied code already works correctly
  - No time to be efficient
- Research shows that 5-10% of large scale computer programs are clones (Baxter, 98)



# Clones in Source Code

- *Dominant decomposition*: A block of statements that performs a function/concern dominates another block
  - The two concerns crosscut each other
  - One concern will have to yield to the other
  - Related to Aspect Oriented Programming (AOP)

# Clones in Source Code



- logging in org.apache.tomcat
  - red shows lines of code that handle logging
  - not in just one place
  - not even in a small number of places

# Clone Dilemma

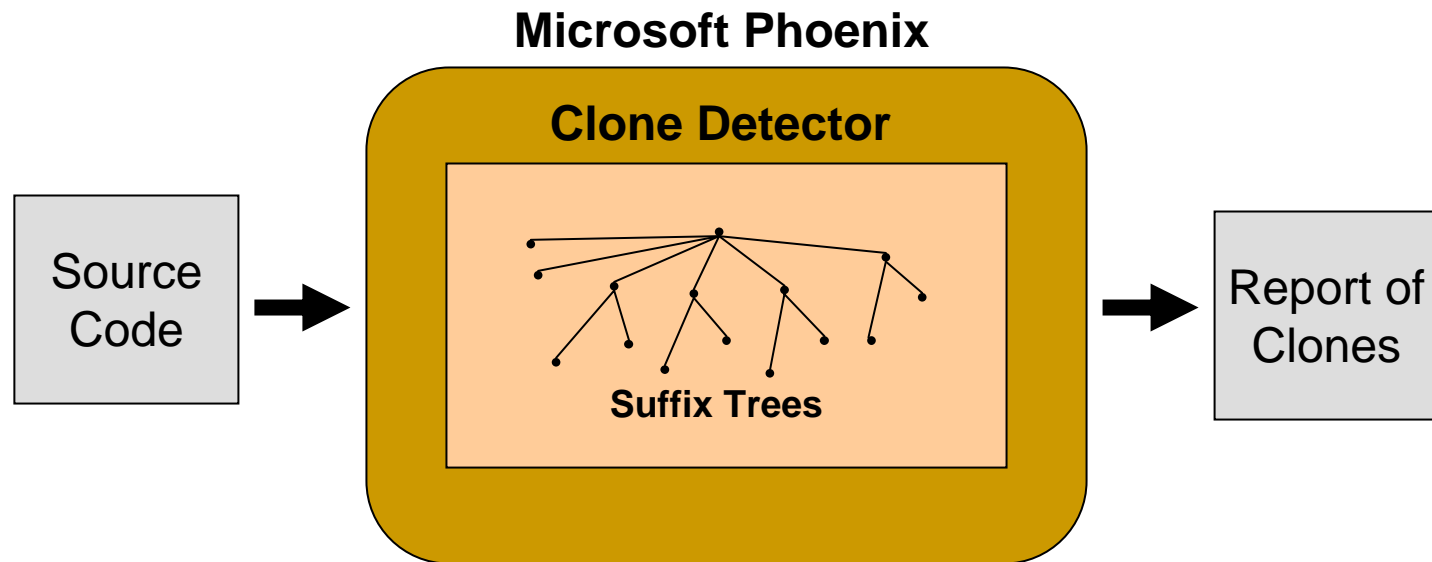
- Maintenance
  - To update code that is cloned will require all clones to be updated
- Restructure/refactor
- Separate into aspects

But first we need  
to find the clones



# Contribution: Automated Clone Detection

- Searches for exact matching function level clones utilizing suffix tree structures in the Microsoft Phoenix framework



# Types of Clones

## Original code

```
int main() {  
    int x = 1;  
    int y = x + 5;  
    return y;  
}
```

```
int func1() {  
    int x = 1;  
    int y = x + 5;  
    return y;  
}
```

Exact match

```
int func2() {  
    int p = 1;  
    int q = p + 5;  
    return q;  
}
```

Exact match, with  
only the variable  
names differing

```
int func3() {  
    int s = 1;  
    int t = s + 5;  
    s++;  
    return t;  
}
```

Near exact match

# What is Phoenix?

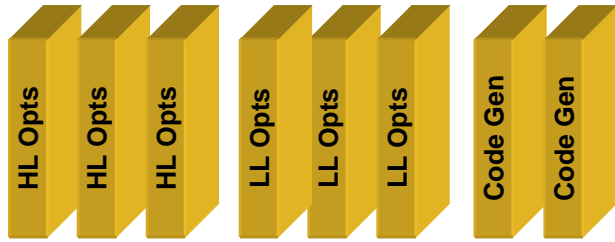
- Next-Generation Framework for
  - building Compilers
  - building Software Analysis Tools
- Basis for Microsoft compilers for 10+ years

More information:

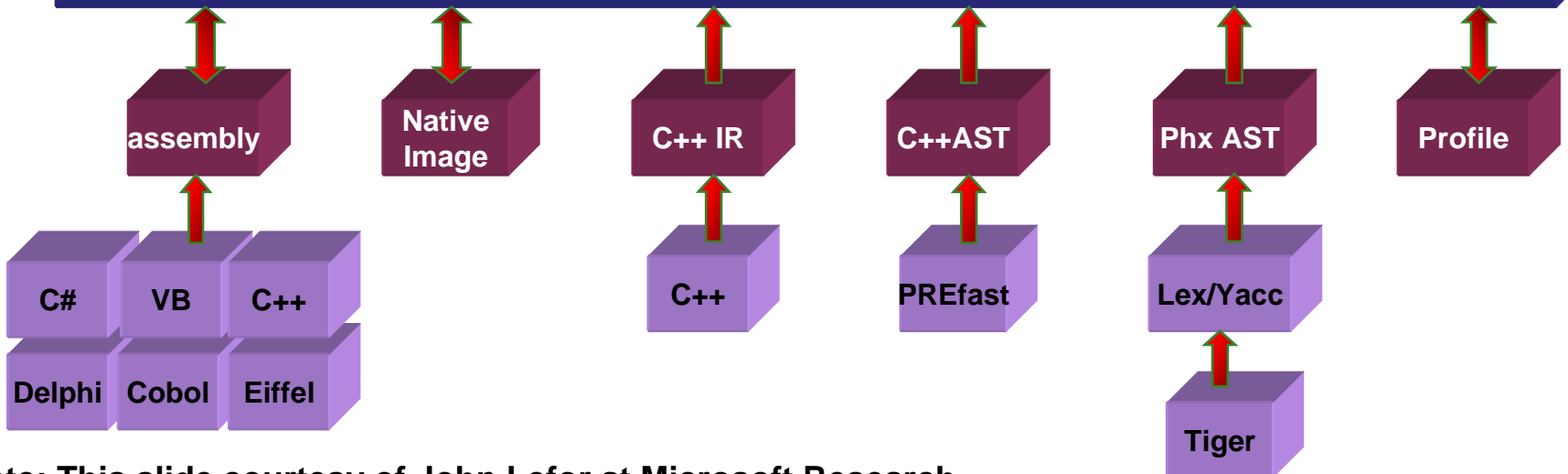
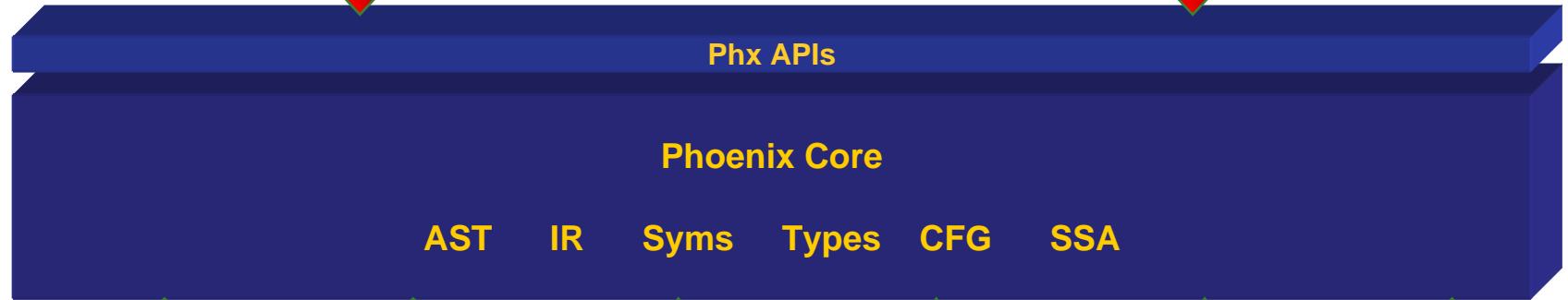
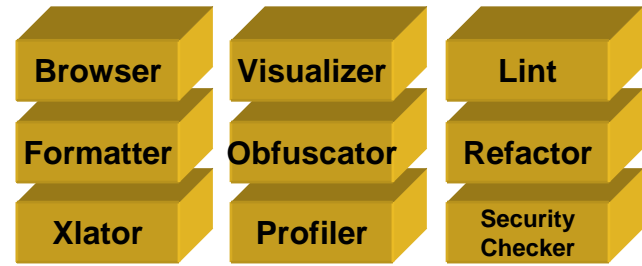
<http://research.microsoft.com/phoenix>



# Compilers



# Tools

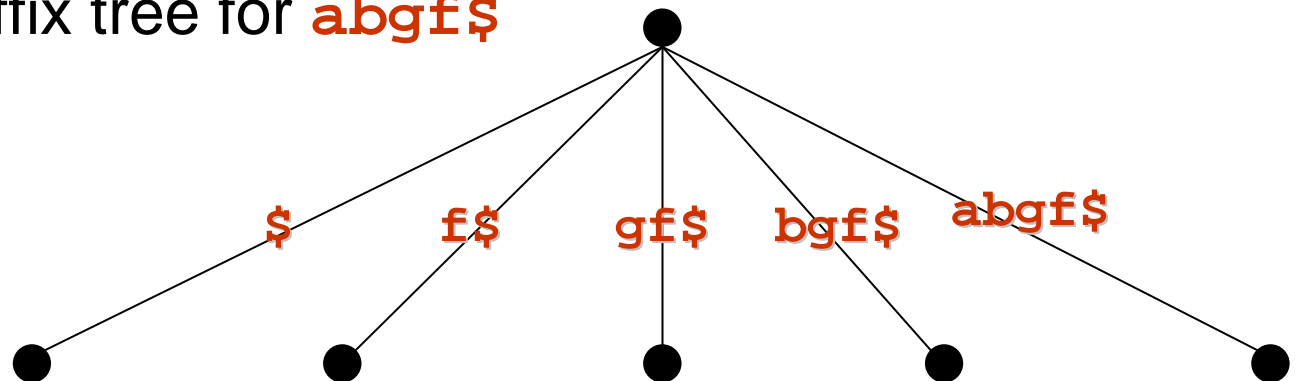


Note: This slide courtesy of John Lefor at Microsoft Research

# Suffix Trees

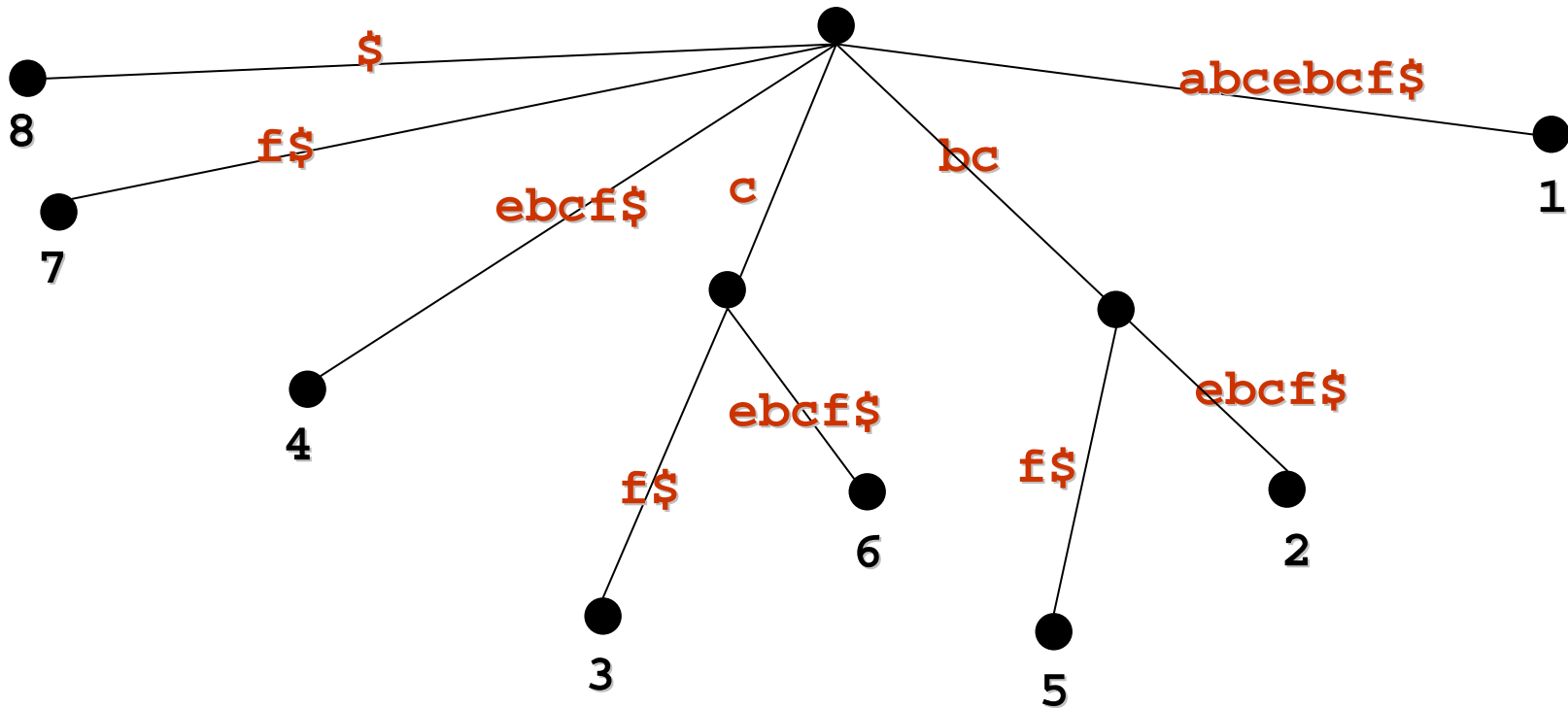
- A suffix tree *of a string* is a tree where each suffix of the string is represented by a path from the root to a leaf
- In bioinformatics it is used to search for patterns in DNA or protein sequences

Example: suffix tree for **abgf\$**



# Another Suffix Tree Example

Suffix tree for **abcebcf\$**  
12345678

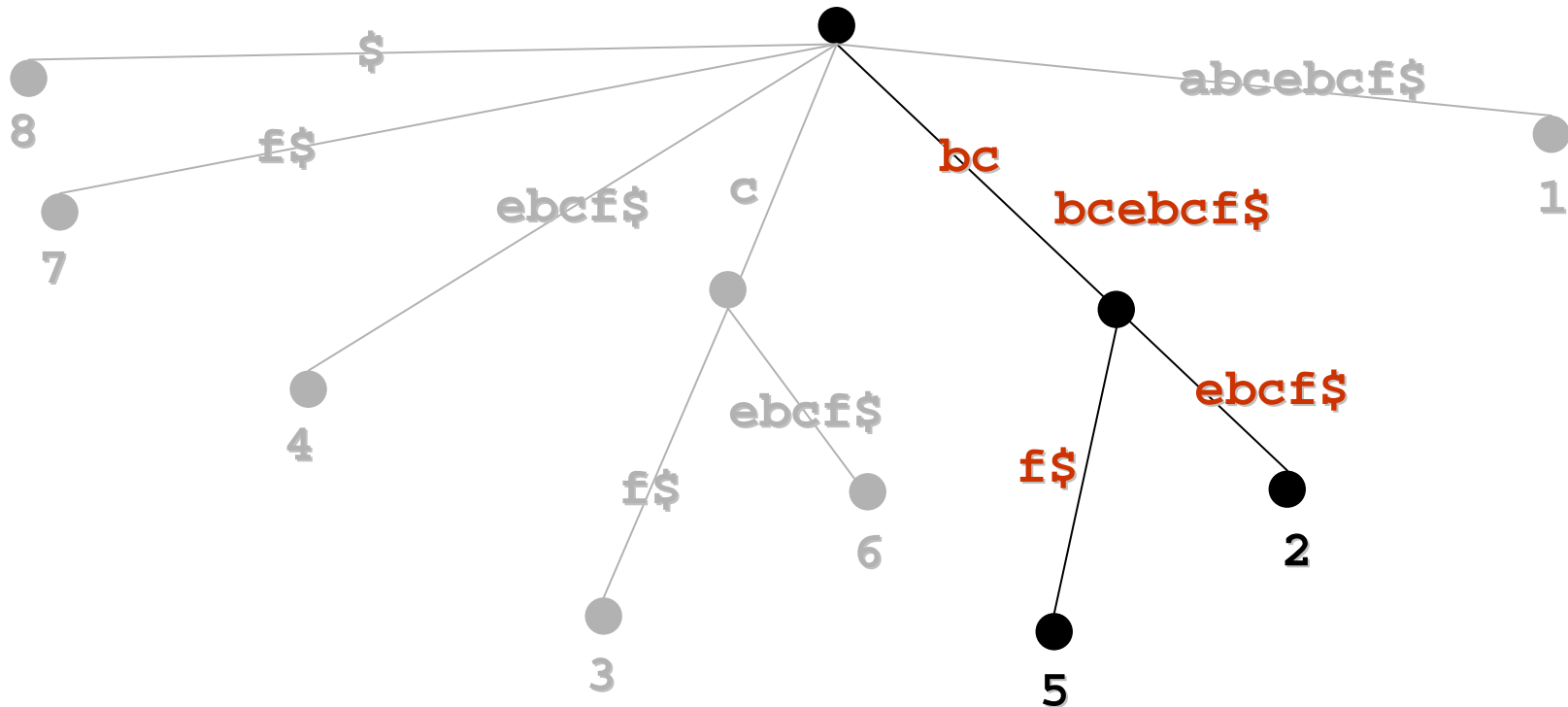


Leaf numbers:

The number indicates the starting position of the suffix from the left of the string.

# Another Suffix Tree Example

Suffix tree for **abcebcf\$**  
12345678



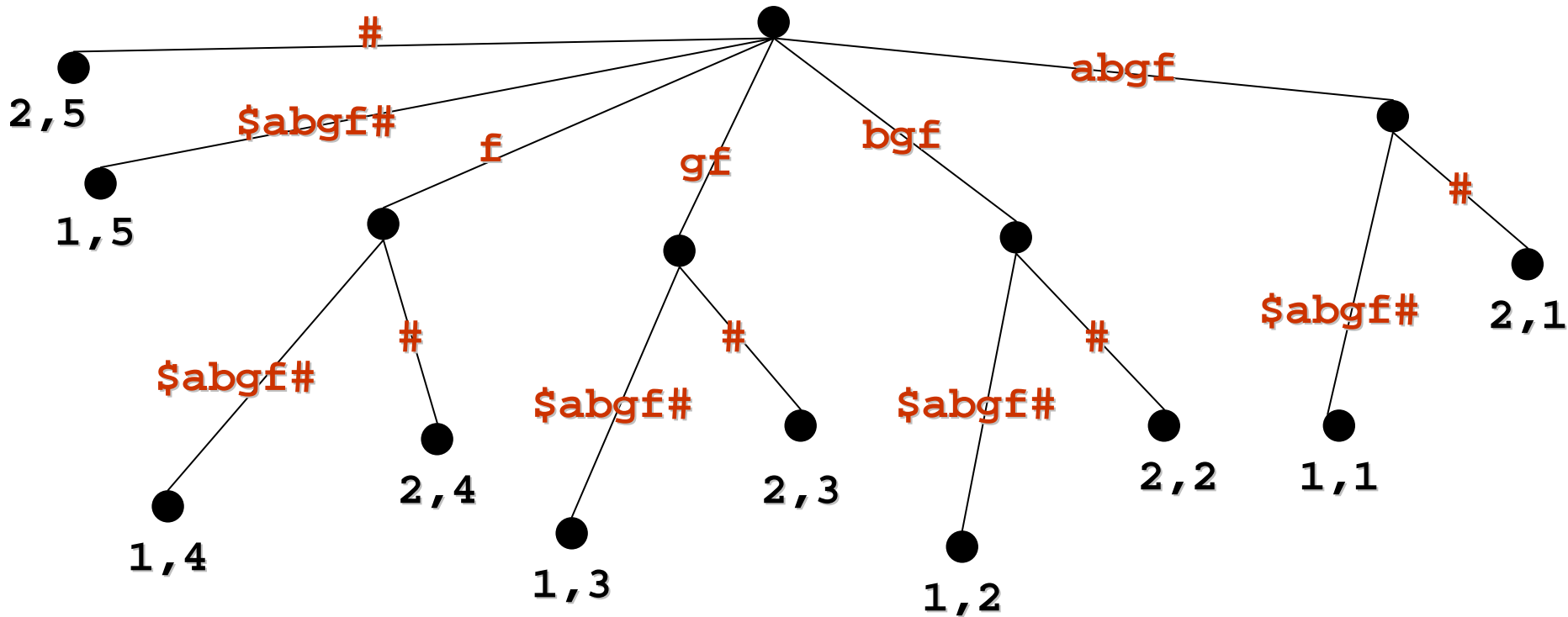
Leaf numbers:

The number indicates the starting position of the suffix from the left of the string.

# Another Suffix Tree Example

Suffix tree for **abgf\$abgf#**

Two identical strings (abgf) separated by unique terminating characters



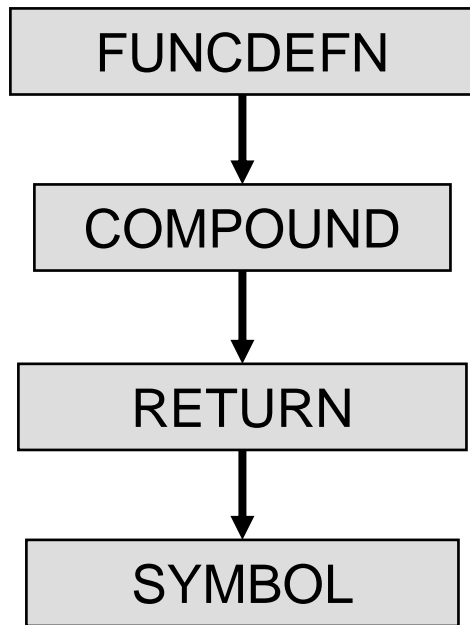
Leaf numbers:

The first number indicates the string.

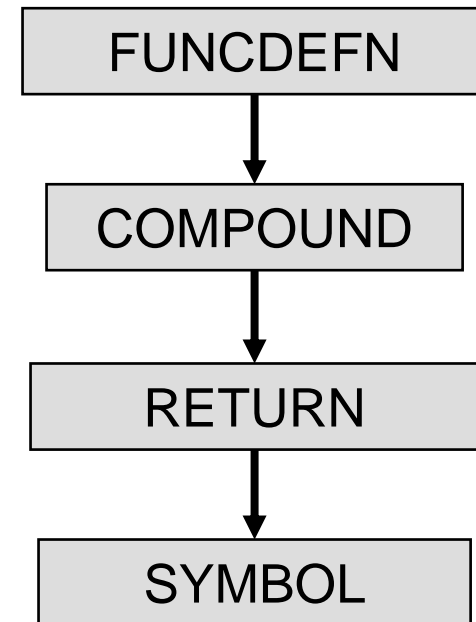
The second number indicates the starting position of the suffix in that string.

# Abstract Syntax Tree Nodes

```
int func1() {  
    return x;  
}
```



```
int func2() {  
    return y;  
}
```

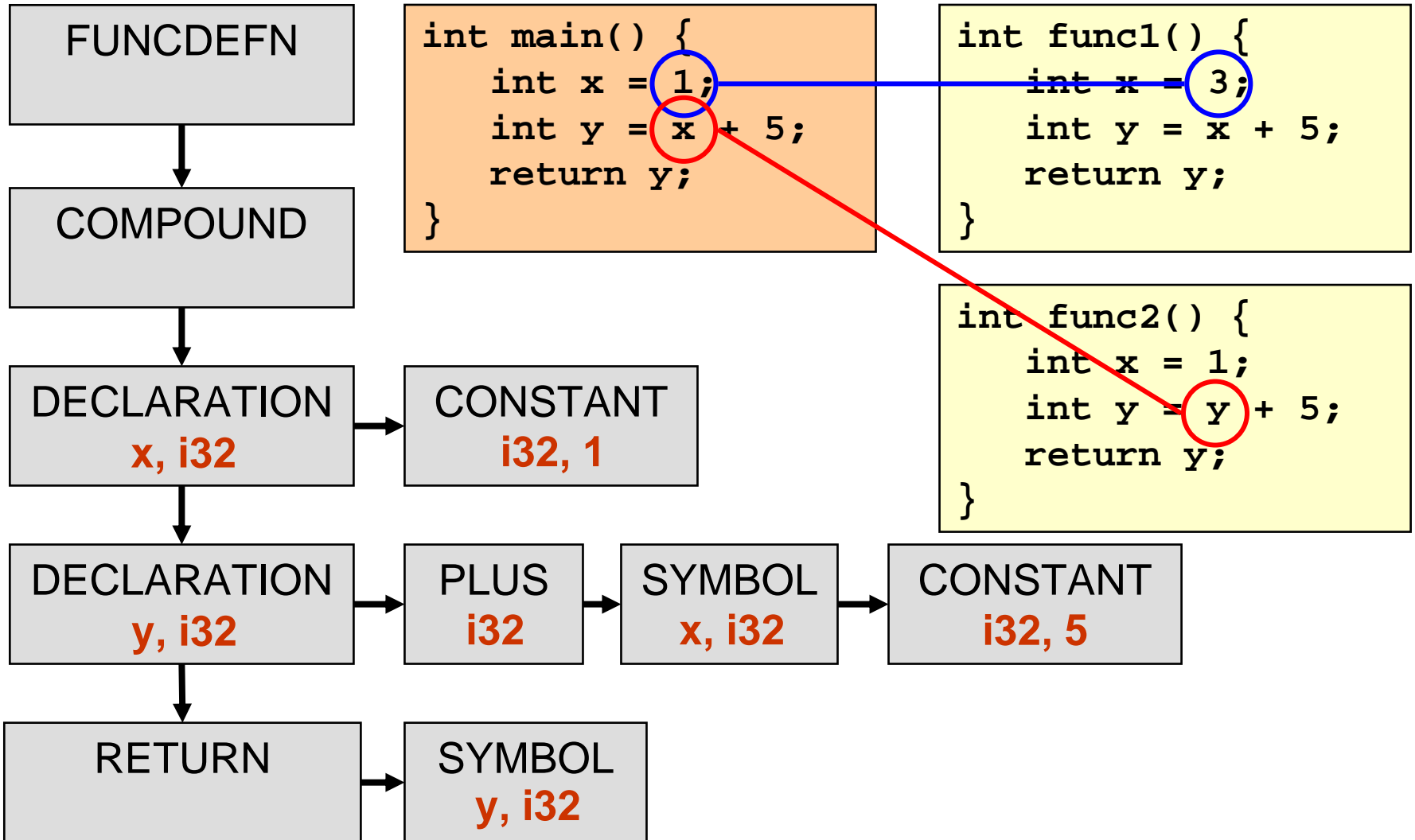


**Note: Node names are Phoenix-defined.**



# False Positives

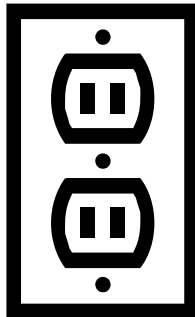
## Original code



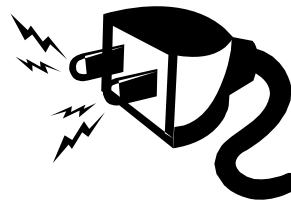
# Phoenix Phases

- Processes are divided into “phases”
- Custom phases can be inserted to perform tasks such as software analysis
- Phases are inserted through “plug-ins” in the form of a library (DLL) module

**Microsoft  
Phoenix**



**Plug-in**

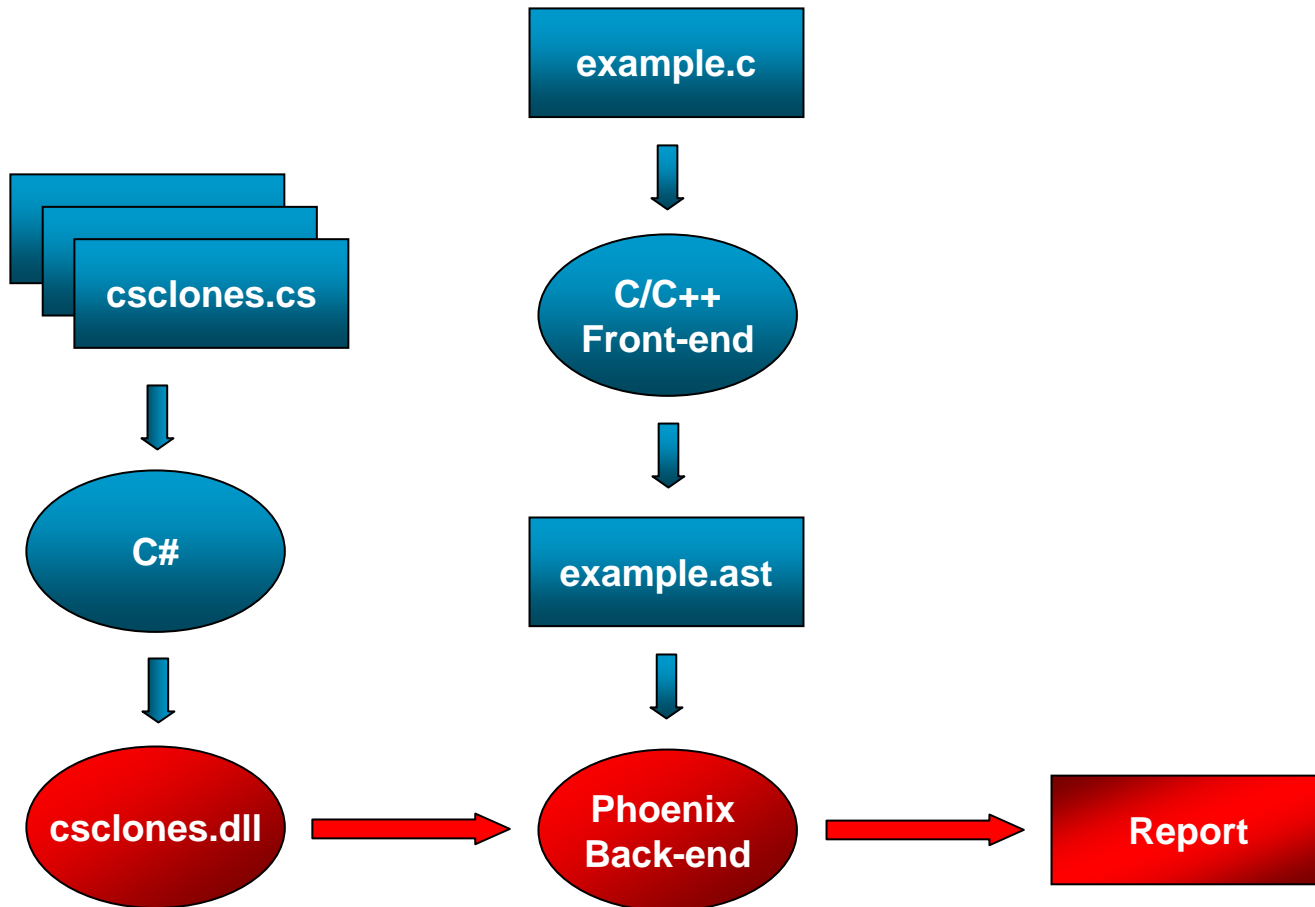


**Custom Phase**

**Clone Detection  
Phase**



# Clone Detector in Phoenix



# Case Study

Program:	Abyss Small web server (~1500 LOC)	Weltab Election results program (~11K LOC)
Duplicate function groups:	<p>Functions <b>ConfGetToken</b> (in conf.c) and <b>GetToken</b> (in http.c).</p> <p>Functions <b>ThreadRun</b> (in thread.c) and <b>ThreadStop</b> (in thread.c).</p> <p>Note: Out of <b>5</b> duplicate function groups found, <b>3</b> are in predefined header files.</p>	<p>Functions <b>ConfGetToken</b> (in conf.c) and <b>GetToken</b> (in http.c).</p> <p>Function <b>canvw</b> in files canv.c, cnv1.c, and cnv1a.c</p> <p>Function <b>lhead</b> in files lans.c and lansxx.c and function <b>rshhead</b> in files r01tmp.c, r101tmp.c, r11tmp.c, r26tmp.c, r51tmp.c, rsum.c, and rsumxx.c</p> <p>Function <b>rsprtpag</b> in files r01tmp.c, r101tmp.c, r11tmp.c, r26tmp.c, r51tmp.c, and rsum.c</p> <p>Function <b>askchange</b> in files vedt.c, vfix.c, and xfix.c</p> <p>Note: Out of <b>6</b> duplicate function groups found, <b>2</b> are in predefined header files.</p>

# Case Studies

- Abyss  
Small web server (~1500 LOC)

Functions <i>ConfGetToken</i> (in conf.c) and <i>GetToken</i> (in http.c).
Functions <i>ConfGetToken</i> (in conf.c) and <i>GetToken</i> (in http.c).

# Limitations and Future Work

- Looks only for exact matches
  - Currently working on a process called hybrid dynamic programming, which includes the use of suffix trees (*k-difference inexact matching*)
- Looks only at the function level
  - Enable multiple levels clone detection
  - Higher: statement level; Lower: program level
- Recognizes only C nodes
  - Coverage for other languages, such as C++ and C#
  - Another approach: language independent

# Thank you...Questions?

## Phoenix-Based Clone Detection using Suffix Trees

<http://www.cis.uab.edu/tairasr/clones>



Department of Computer and Information Sciences

University of **A**labama at **B**irmingham