Lightweight techniques for tracking unique program statements

Jaime Spacco, Colgate University
jspacco@colgate.edu
Chadd Williams, Pacific University
chadd@pacificu.edu

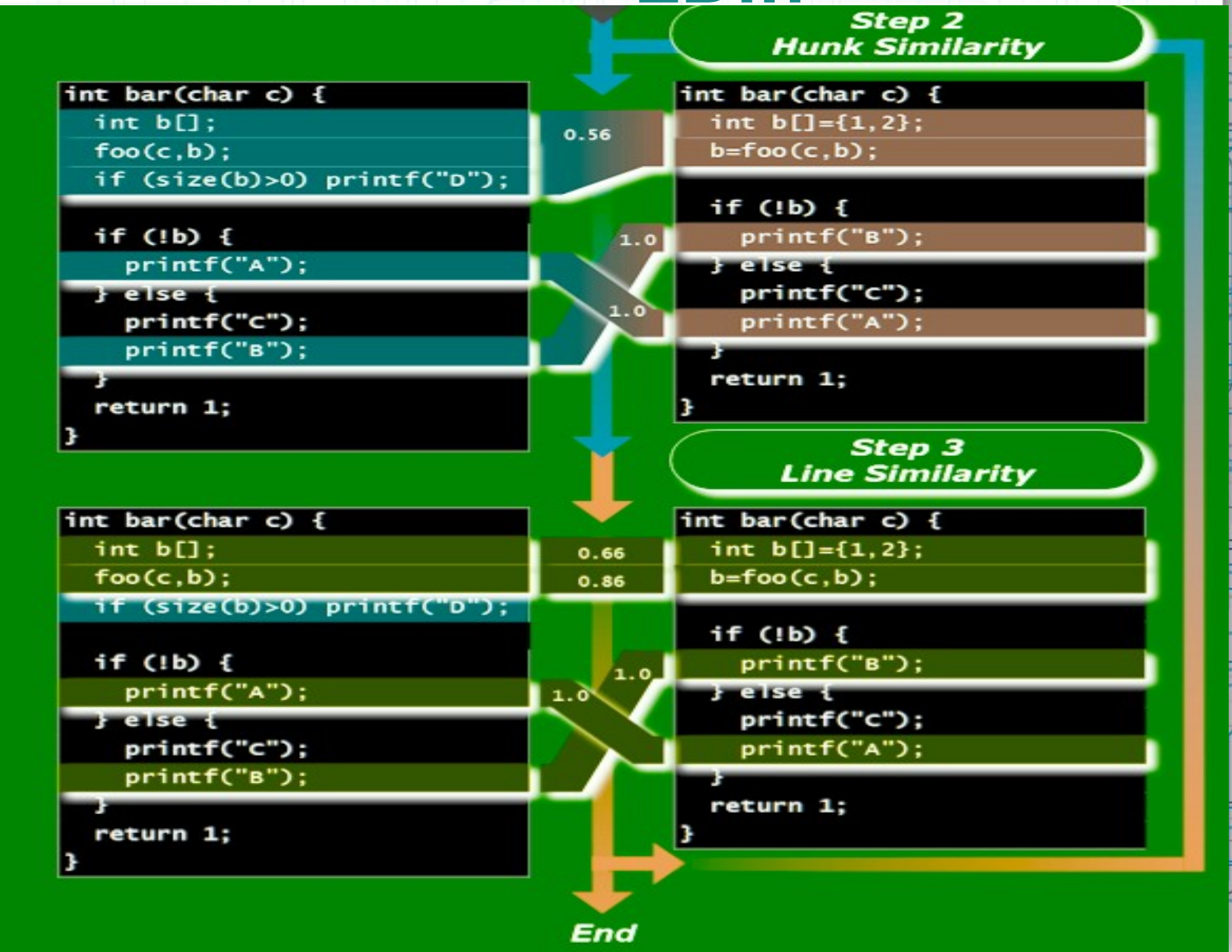
The problem

- Given two versions of a file...
  - track the ancestry of a line as accurately as possible
- Possible uses:
  - Identifying exactly where bugs are introduced
  - Studying software evolution
  - Code clone detection

Best available solutions

- Canfora et al.
  - LDiff (line difference)
  - MSR 2007, ICSE 2009
- Reiss
  - large suite of techniques
  - ICSE 2008
- Our work
  - SDiff (statement difference)

LDiff


Step 2
Hunk Similarity
int bar(char c) {
int b[];
foo(c,b);
if (size(b)>0) printf("D");
if (!b) {
printf("A");
} else {
printf("C");
printf("B");
}
return 1;
}
0.56
1.0
1.0
int bar(char c) {
int b[]={1,2};
b=foo(c,b);
if (!b) {
printf("B");
} else {
printf("C");
printf("A");
}
return 1;
}
Step 3
Line Similarity
int bar(char c) {
int b[];
foo(c,b);
if (size(b)>0) printf("D");
if (!b) {
printf("A");
} else {
printf("C");
printf("B");
}
return 1;
}
0.66
0.86
1.0
1.0
int bar(char c) {
int b[]={1,2};
b=foo(c,b);
if (!b) {
printf("B");
} else {
printf("C");
printf("A");
}
return 1;
}
End

Reiss

- Huge suite of techniques
  - Combination of simple techniques worked best on his test suite
- Essentially, a weighted combination of:
  - line similarity metric (Levenshtein)
  - context similarity metric for surrounding lines
- More complex techniques available
  - Reiss's ICSE 2008 paper lists them all
  - didn't perform as well as hybrids of simple techniques

Our contribution: SDiff

- *Statement Diff*
- Diff Java statements rather than lines of text

Finding statements

- Parse Java code to an AST
  - use parser from PMD (static style checker)
- Break the AST into statements
- Turn the statements into a canonical tokenized format
- match the statements between the two versions

SDiff

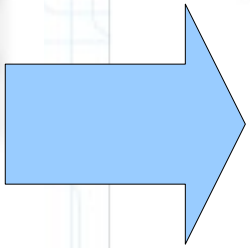
```
public int f(int x) {
   int z=g(x);
   int w = getW();
   int result=p(z, w, x);
   return result; }
```

```
public int f(int x)
{
   int z=g(x);
   int w = getW();
   // compute result
   int result=p(z,
         w,
         x);

   return result;
}
```

Matching statements

```
public int f(int x) {
    int z=g(x);
    int w = getW();
    int result=p(z, w, x);
    return result; }
```



```
public int f(int x)
{
    int z=g(x);
    int w = getW();
    // compute result
    int result=p(z,
        w,
        x);
    return result;
}
```

Advantages of SDiff

- Automatically ignores whitespace, comments and brackets
- Handles statements broken across multiple lines
- Can diff by character or by token

Contributions:

- SDiff algorithm
  - http://code.google.com/p/sdiff/
- Running best available algorithms:
  - on Reiss's test suite
  - on a new test suite

Reiss's test suite

- Track 53 lines across 25 revisions of one source file
- all changes are single lines
  - includes adds, deletes and changes
- Only two pairs of adjacent lines

Our test suite

- 20 files from the Eclipse version control repository
  - before and after an edit
  - each edit is 3-12 lines

Reiss's test suite

Algorithm	Recall %
Reiss w_besti_linei_method	97.7
Reiss w_besti_besti	97.0
LDiff tweaked	85.0
LDiff default	83.0
SDiff	86.2
SDiff, ignore non-executable statements	96.1

Eclipse test suite

Algorithm	Recall %
Reiss w_best_diff	66
Reiss w_smart_best	61
LDiff tweaked	64
LDiff default	45
SDiff tweaked	70
SDiff default	66

Controversial Statement

- You can't match up individual lines between large diff chunks
  - You can only match changes up to N lines
- Some changes shouldn't even have an ancestry
  - a 1-character edit distance to a for loop header is different if the 40 surrounding lines have also changed
- Code available:
  - http://code.google.com/p/sdiff/

Big questions

- When is it useful to track lines backwards?
  - do we care about the ancestry of a curly brace?
- How large can diff chunks be before we lose track of them?

Strengths and weaknesses of each approach

Tasks	LDiff	Reiss	SDiff
Match comments	yes	yes	no
detect edited lines	yes	yes	yes
handle any arbitrary text file	yes	yes	no
Use context of surrounding lines to assist matching	yes	yes	yes
supported unordered similarity metrics	yes	no	no
detect renamed methods	no	no	yes
detect statements broken across several lines	no	no	yes
easy to download and run	yes	no	yes

Strengths and weaknesses of each approach

Tasks	LDiff	Reiss	SDiff
Match comments	yes	yes	no
detect edited lines	yes	yes	yes
handle any arbitrary text file	yes	yes	no
Use context of surrounding lines to assist matching	yes	yes	yes
supported unordered similarity metrics	yes	no	no
detect renamed methods	no	no	yes
detect statements broken across several lines	no	no	yes
easy to download and run	yes	no	yes

http://code.google.com/p/sdiff/