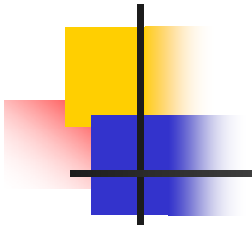
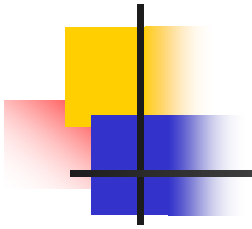


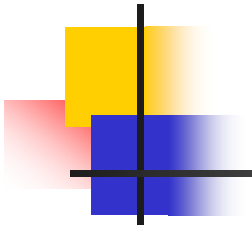
Motivation

- Underlying question : *How does software change ?*
  - In : Two versions of a program
  - Out : Picture of changes
- Relevance
  - Software development
  - Software engineering



Objective and Approach

- Summarize C program changes
  - Functions (body AST, prototype)
  - Global variables (type and initializer)
  - Types
    - Structs/Unions (fields deleted / added / type changed)
    - Typedefs
    - Enums
- Our Approach: AST matching
  - Accurate; handles renamings
  - Scales to real-world applications; e.g., Apache, Linux kernel, OpenSSH



Raw Output

struct "net_device": 1 fields changed type: “accept_fastpath”
struct "reiserfs_journal": 1 fields deleted: “j_dummy_inode”
struct "reiserfs_journal": 1 fields added: “j_dirty_buffers”

function "block_read_full_page": 1 arguments changed type: “get_block”
function "ext2_readdir": 1 arguments changed type: “filldir___0”

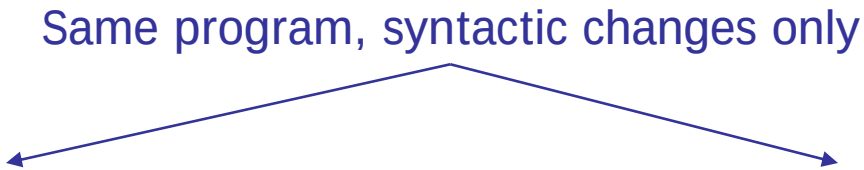
+ function “inetdev_changename”
+ function “__ide_dma_good_drive”
+ function “ide_unplugged_outbsync”
+ function “inode_init_once”
- function “target_cpus”
- function “ide_dmafunc_verbose”

+ typedef “cisco_proto”
- typedef “ide_ioctl_proc”

+ global var “idecd”

Linux 2.4.20 vs 2.4.21

The Renaming Problem



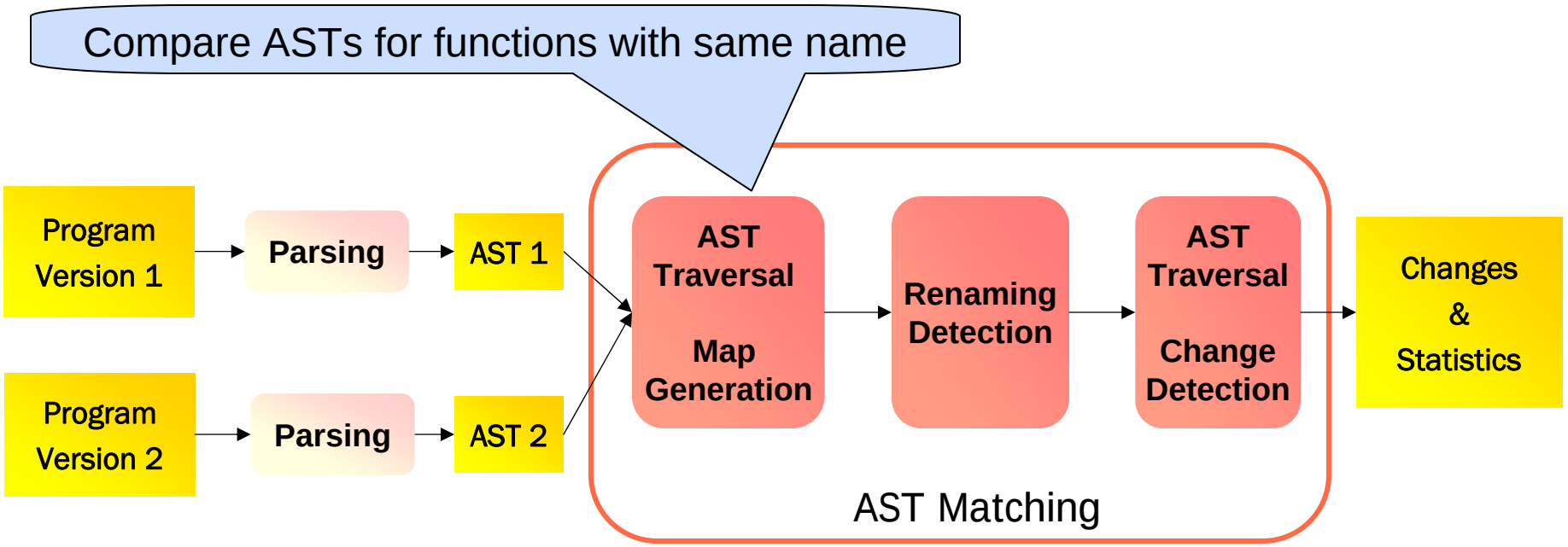

```
typedef int sz_t;

struct foo {
  int i;
};

int count;

void f(int a) {
   struct foo sf;
   sz_t c = 2;
   sf.i = a + c;
   count++;
}
```

Version 1

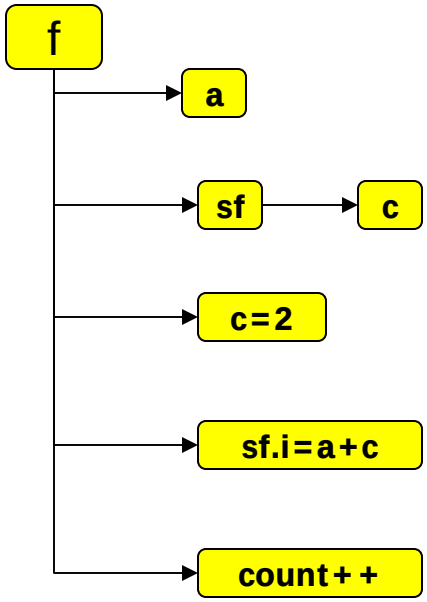
```
typedef int size_t;

struct bar {
  int i;
};

int counter;

void f(int b) {
   struct bar sb;
   size_t d = 2;
   sf.i = b + d;
   counter++;
}
```

Version 2

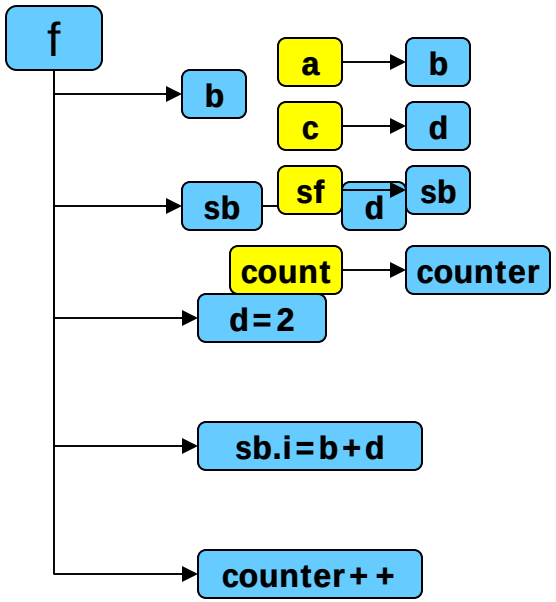
Abstract Syntax Tree Matching



AST Traversal - Name Map Generation



Name Map



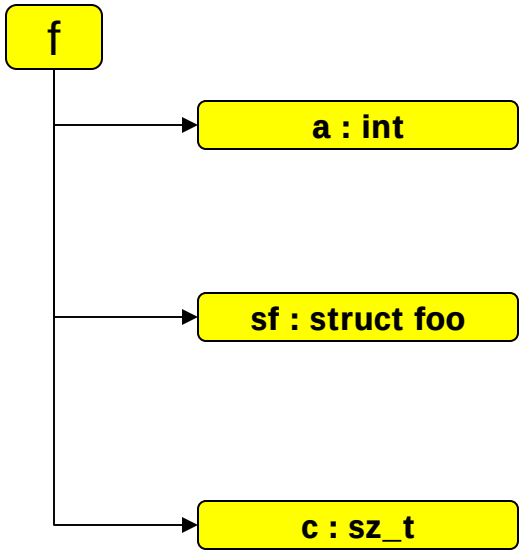

```
void f(int a) {
   struct foo sf;
   sz_t c = 2;
   sf.i = a + c;
   count++;
}
```

Version 1

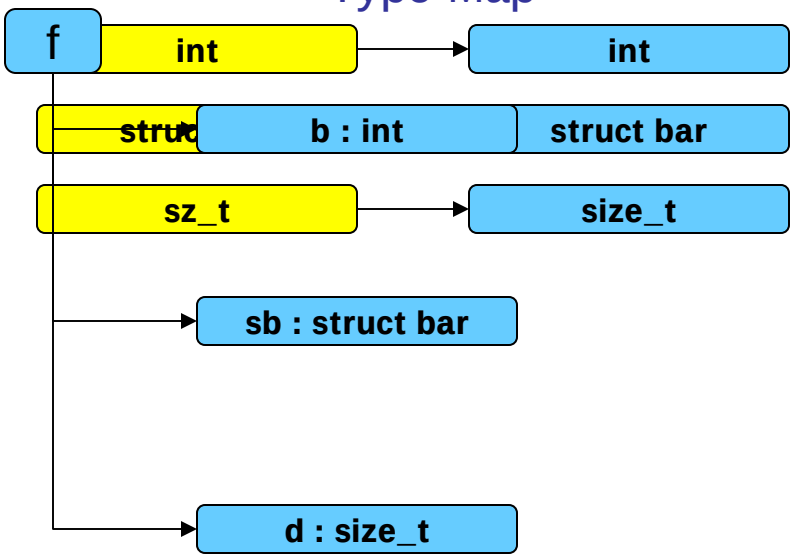
```
void f(int b) {
   struct bar sb;
   size_t d = 2;
   sf.i = b + d;
   counter++;
}
```

Version 2

AST Traversal - Type Map Generation



Type Map



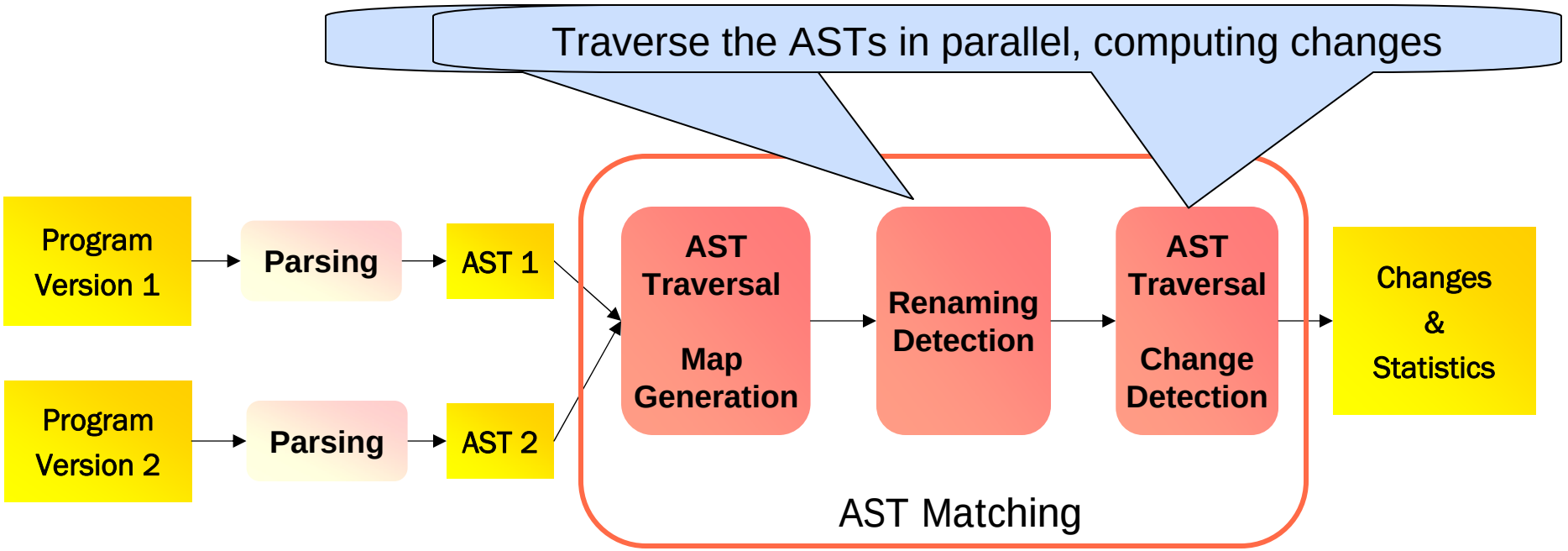

```
void f(int a) {
   struct foo sf;
   sz_t c = 2;
   sf.i = a + c;
   count++;
}
```

Version 1

```
void f(int b) {
   struct bar sb;
   size_t d = 2;
   sf.i = b + d;
   counter++;
}
```

Version 2

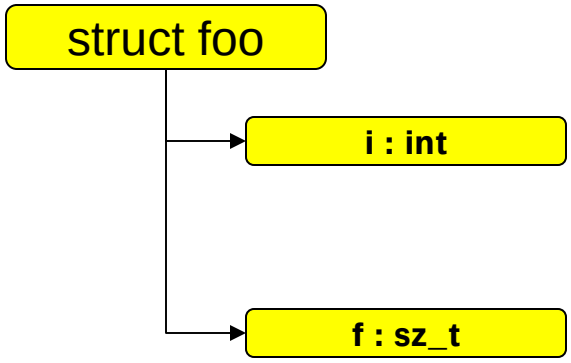
Abstract Syntax Tree Matching

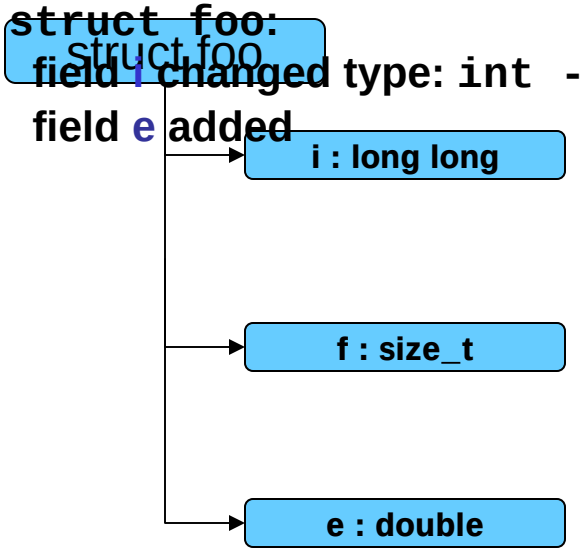


A renamed to B iff

- A → B in the map
- A deleted
- B added

AST Traversal - Change Detection





struct foo:
field i changed type: `int -> long long`
field e added

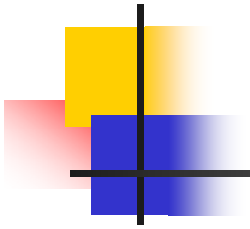
```
typedef int sz_t;
struct foo {
   int i;
   sz_t f;
}
```

Version 1

sz_t → size_t

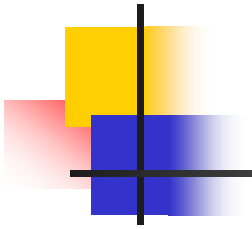
```
typedef int size_t;
struct foo {
   long long i;
   size_t f;
   double e;
}
```

Version 2



Implementation

- Parsing via CIL toolkit
  - Merges whole program into single, preprocessed file
- Fast
  - Scales linearly, 400.000 LOC in 1 minute
- Generates different output formats
  - Raw differences, summaries, density trees



Summary Statistics

------- Functions -------
Version1 : 7697
Version2 : 7881
added : 232
deleted : 48
locals/formals changed name : 3
arguments type changes : 19
return types changes : 15

------- Structs/Unions -------
Version1 : 1214
Version2 : 1233
added : 17
deleted : 1
field type changes : 15
field count changes : 19

------- Typedefs -------
Version1 : 487
Version2 : 469
added : 13
deleted : 31
base type changes : 2

------- Global Variables ---
Version1 : 8027
Version2 : 8074
added : 43
deleted : 16
var type changes : 11
var val changes : 51

------- Enums -------
Version1 :33
Version2 : 31
deleted : 2
item count changes : 1
var exp changes : 20

Linux 2.4.20 vs 2.4.21

Density Trees

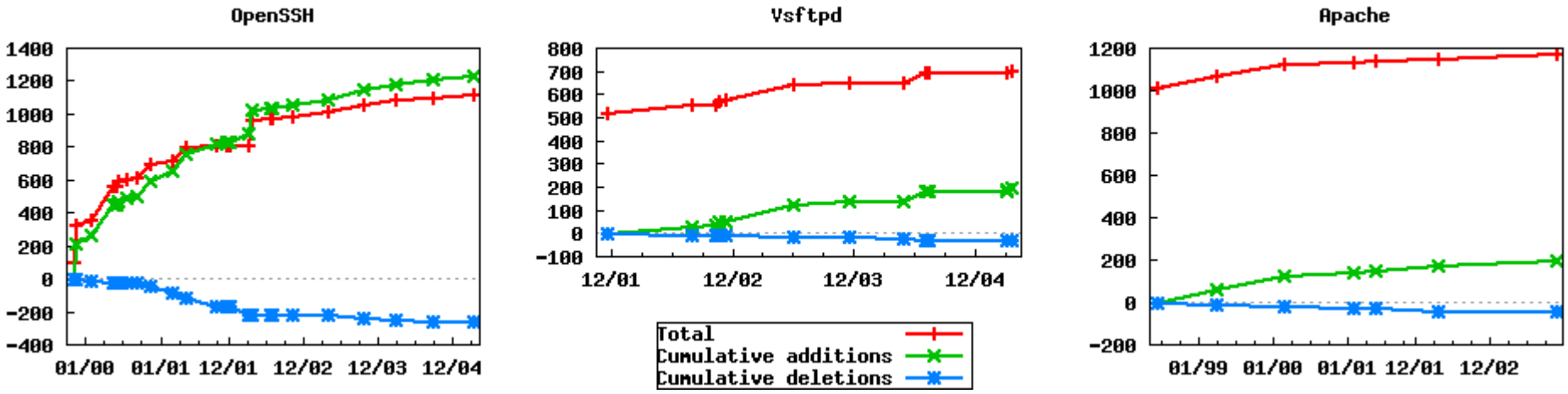
```
/ : 111
  include/ : 101
    linux/ : 96
      fs.h : 4
      ide.h : 80
      reiserfs_fs_sb.h : 1
      reiserfs_fs_i.h : 2
      sched.h : 1
      wireless.h : 1
      hdreg.h : 7
    net/ : 2
      tcp.h : 1
      sock.h : 1
    asm-i386/ : 3
      io_apic.h : 3
  drivers/ : 9
    char/ : 1
      agp/ : 1
        agp.h : 1
    ide/ : 8
      ide-pci.c : 8
  net/ : 1
    ipv4/ : 1
      ip_fragment.c : 1
```

Struct/Union field additions

Linux 2.4.20 vs 2.4.21

Case Studies: OpenSSH, Vsftpd, Apache

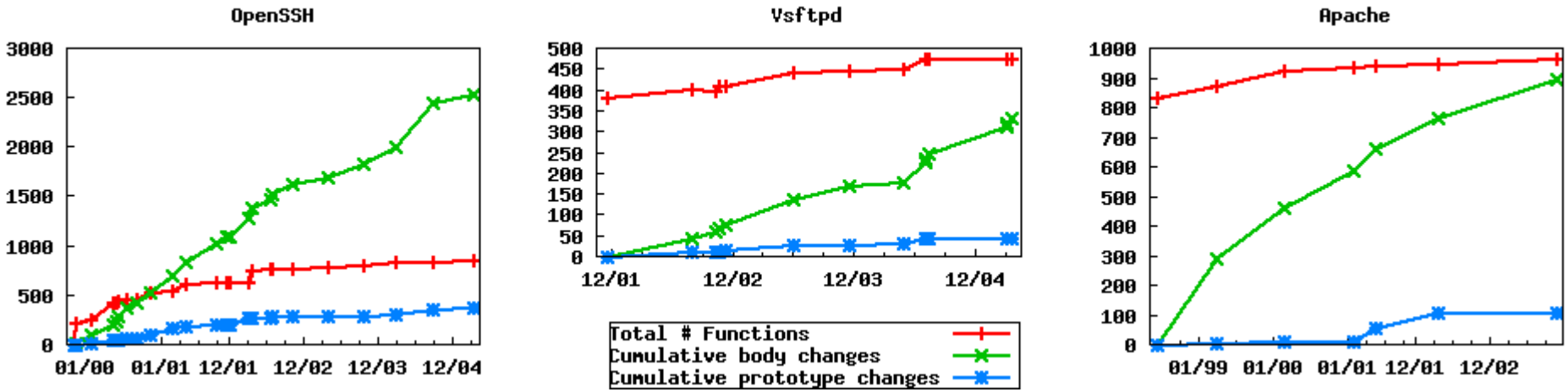
Functions & global variables: how often added and deleted?



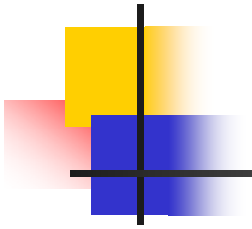
- OpenSSH changes most frequently
- Deletions infrequent, relative to additions

Case Studies: OpenSSH, Vsftpd, Apache

How often do function bodies and prototypes change?

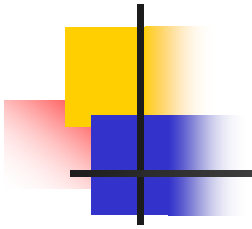


- Function bodies do change a lot
- Function prototypes do not change much



Related Approaches

- Standard `diff`
  - Low-level
  - Verbose: Linux 2.4.20->2.4.21 patch : 21MB
- Release notes
  - High level
  - Possibly incomplete



Summary

- Approach for reporting changes to C programs
  - AST-matching
  - Variety of changes at several levels of detail
  - Accurate
  - Scalable
- Soon to be available at http://www.cs.umd.edu/~neamtiu/evolution